Daniel Filipe Soares Silva

Soluções IoT para Caracterização de Tráfego em Cidades Inteligentes

IoT solutions for Traffic Characterization in Smart Cities

"*The mind that opens to a new idea never returns to its original size.*"

— Albert Einstein

Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

Daniel Filipe Soares Silva

Soluções IoT para Caracterização de Tráfego em Cidades Inteligentes

IoT solutions for Traffic Characterization in Smart Cities

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor João Paulo Silva Barraca, Professor auxiliar convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Diogo Nuno Pereira Gomes, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Este trabalho insere-se no âmbito do projeto @CRUiSE (PTDC/EMS-TRA/0383/2014), financiado no âmbito do Projeto 9471 – Reforçar a Investigação, o Desenvolvimento Tecnológico e a Inovação (Projeto 9471 – RIDTI) e comparticipado pelo Fundo Comunitário Europeu FEDER, e no âmbito do Projeto Estratégico UID-EMS-00481-2013.

o júri / the jury

presidente / president

Professor Doutor André Ventura da Cruz Marnoto Zúquete
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professora Doutora Margarida Isabel Cabrita Marques Coelho
Professora Auxiliar da Universidade de Aveiro

Professor Doutor João Paulo Barraca
Professor Auxiliar da Universidade de Aveiro

agradecimentos /
acknowledgements

Em primeiro lugar, quero agradecer ao meu orientador, Professor Doutor João Paulo Barraca, por todo o seu apoio e orientação, fulcral, prestado ao longo da realização desta Dissertação e também ao Professor Doutor Diogo Gomes, pela sua ajuda na realização deste trabalho.

Quero do mesmo modo manifestar o meu apreço ao grupo ATNoG e ao Instituto de Telecomunicações, por toda a disponibilidade em prestar as melhores condições de trabalho.

A todos os envolvidos no projeto @CRUiSE, quero agradecer por toda a ajuda, e desejo todo o sucesso na continuidade do seu trabalho.

Em segundo lugar, quero agradecer à minha família, em particular aos meus pais e ao meu irmão, que sempre me apoiaram durante a minha vida académica.

Um agradecimento especial à Filipa Alves, por ter sido um apoio essencial, constante, incansável, e sempre disponível para me ajudar.

Por último, quero agradecer a todos os meus amigos, que durante estes magníficos anos, partilhámos sucessos e dificuldades. Em particular aos meus amigos mais chegados, um muito obrigado André Santos, Diogo Cardoso, Ivo Silva, Guilherme Cardoso, Tiago Magalhães, Tiago Oliveira, Ricardo Martins e Rui Pedro.

Palavras Chave

Internet das Coisas, Controlo de tráfego, Cidades Inteligentes, Registo de dados.

Resumo

As agências que administram o tráfego rodoviário têm de tomar decisões importantes, a fim de definir quais as secções de estrada que têm o maior risco de impactes relacionados com o tráfego. Neste contexto, reconhece-se que a implementação do Sistema Avançado de Gestão de Tráfego (SAGT) pode melhorar não apenas a eficiência da rede, mas também minimizar outras externalidades de tráfego. Neste contexto, novas soluções de software e hardware, capazes de fornecer informações melhoradas a partir da dinâmica do tráfego, podem desempenhar um papel essencial no conhecimento que temos e a forma como o SAGT é executado. Em particular, a disponibilidade de dados georreferenciados está a aumentar rápidamente, seja em dispositivos móveis, como em redes sociais e monitoramento de redes de sensores. Um dos desafios é combinar e melhorar o potencial de cada fonte de informação, e depois juntar várias fontes num modelo agregado. Nesta dissertação é descrita a arquitetura e implementação de diferentes dispositivos para recolha de dados, um protótipo para monitorização com integração de parâmetros do motor em tempo-real e uma aplicação móvel, o desenvolvimento de toda a infra-estrutura necessária e ainda uma aplicação web que combine estes dados e forneça ferramentas de análise e visualização. Nesta dissertação é possível que os utilizadores possam utilizar estas ferramentas para adotarem escolhas mais sustentáveis e uso de estradas menos congestionadas, contribuindo para a diminuição do congestionamento do tráfego, poupando tempo, aumentando o fluxo de tráfego, e contribuindo positivamente para o impacte ambiental.

Keywords Internet of Things, Traffic control, Smart Cities, Data logging.

Abstract

Agencies managing road traffic need to make informed decisions, in order to define which road sections have the highest risk of traffic-related impacts. In this context, it is recognized that the implementation of Advanced Traffic Management System (ATMS) may improve not only network efficiency but also minimize other traffic externalities. In this context, novel software and hardware solutions, capable of providing improved information from the traffic dynamics, can play an essential role in the knowledge we have, and the way ATMS is executed. In particular, the availability of geo-referenced data is increasing quickly, either from nomadic devices as well as from social media, and monitoring sensors networks. One of the challenges is to combine and to improve the potential of each source of information, and then combine multiple sources together in an aggregate model. This dissertation describes the architecture and implementation of an accurate, high-frequency vehicle tracker, with the integration of real-time engine statistics, and enhanced with an autonomous inertial model as well as a mobile application for data collection from the embedded sensors and positioning, the development of all necessary infrastructure and a web application that combine these data and provide analysis and visualization tools. In this dissertation it is possible for users to use these tools to adopt more sustainable choices and use of less congested roads, contributing to the reduction of traffic congestion, saving time, improving traffic throughput, and contributing positively to the environmental impact.

Contents

List of Figures

List of Tables

Acronyms

3GPP	3rd Generation Partnership Project
6LoWPAN	Low power Wireless Personal Area Network
ADC	Analog-to-Digital Converter
AM	Amplitude Modulation
API	Application Programming Interface
APS	Application Support Sublayer
ATMS	Advanced Traffic Management System
BS	Base Station
CAN	Crontoller Area Network
Cassandra	Apache Cassandra database
CATE	Computer-Assisted Traveling Environment
CESAM	Centro de Estudos do Ambiente e do Mar
CF	Complementary Filter
CO	Carbon Monoxide
CO2	Carbon Dioxide
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
@CRUiSE	Advanced Impact Integration Platform for Cooperative Road Use
CR	Cognitive Radio
CSV	Comma-separated Values
DETI	Department of Electronics Telecommunications and Informatics
DOF	Degrees Of Freedom
FCD	Floating Car Data
FM	Frequency Modulation
GIS	Geographic Information System
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Service
GPS	Global Positioning System
GR-GSM	GnuRadio GSM
GSM	Global System for Mobile Communications
GTL	Gas to Liquid
hPa	Hectopascal
HTML5	Hypertext Markup Language version 5
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTP/2	Hypertext Transfer Protocol version 2
I^2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IEETA	Institute of Electronics and Informatics Engineering of Aveiro
IETF	Internet Engineering Task Force
ILD	Inductive Loop Detector
IMT	International Mobile Telecommunications
IMU	Inertial Measurement Unit
IoT	Internet of Things
IPv4	Internet Protocol version 4

IPv6	Internet Protocol version 6
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
IT	Institute of Telecommunications
ITS	Intelligent Transportation System
JSON	JavaScript Object Notation
kPa	kilo Pascal
kms/h	Kilometers per Hour
LAN	Local Area Network
LED	Light Emitting Diode
LLN	Low-Power and Lossy Networks
LMSC	LAN/MAN Standards Commitee
LoRa	Long-Range
LoRaWAN	Low-Power Wide-Area Network
LR-WPAN	Low Rate WPANs
LTE	Long Term Evolution
M2M	Machine-to-Machine
MAC	Medium Access Control
MAF	Mass Air Flow Rate
MAN	Metropolitan Area Network
MAP	Manifold Absolute Pressure
MQTT	Message Queue Telemetry Transport
NFC	Near Field Communication
NIC	Network Interface Cards
NOx	Nitrogen Oxides
OBDII	Updated On-board Diagnostics
OSI	Open Systems Interconnection
OSM	Open Street Map
PANA	Protocol for Carrying Authentication for Network Access
PARADIS	Passive RAdiometric Device Identification System
PDF	Portable Document Format
PNG	Portable Network Graphics
QoS	Quality of Service
RabbitMQ	Open Source Enterprise Messaging
RAM	Random Access Memory
REST	Representational State Transfer
RF	Radio Frequency
RFID	Radio-Frequency IDentification
RPL	Routing Protocol for Low Power and Lossy Networks
RPM	Revolutions per Minute
RTCM	Seminar of the Mobile Communications Thematic Network
SCoT	Smart Cloud of Things
SDR	Software Defined Radio
SNR	Signal-to-Noise Ratio
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SSL	Secure Sockets Layer
STFT	Short Time Fourier Transform
TCP	Transmission Control Protocol
TEMA	Centre for Mechanical Technology and Automation
THC	Total Hydrocarbons
THW	Temperature Humidity Wind Index
TLS	Transport Layer Security
UA	University of Aveiro
UAVs	Unmanned Aerial Vehicles
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
USB	Universal Serial Bus
UUID	Universally Unique Identifier
UV	Ultra-Violet Index
VANET	Vehicular Ad-hoc Networks
VoIP	Voice Over IP
VSP	Vehicle-Specific Power
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSGI	Web Server Gateway Interface
WSN	Wireless Sensor Network
ZCL	ZigBee Cluster Library
ZDO	ZigBee Device Object

CHAPTER 1

Introduction

With the increase in road traffic, drivers continuously experience several high stressful situations: traffic jams, road accidents, pollution, amongst others. These factors are the primary motivation to develop efficient and intelligent ways to help drivers cope with today's traffic. In this document, we are going to explore and describe the current state of the art, some crucial points under the scope of the Advanced Impact Integration Platform for Cooperative Road Use (@CRUiSE) project and an implementation of a solution for this. With this in mind, one of the most important is the impact of technology and environmental information on human behavior and crowdsourcing as a method to gather drivers information, as well as general citizen density to help better planning the public transport network. A considerable amount of research has been made regarding driving behavior analysis, using mobile phones as sensors [39]. This justifies the growing interest in developing a platform that can gather real-time information from different sources, processing it using in-house optimization toolbox and providing relevant information to each driver, enabling them to use safer roads, avoid usually congested roads and to make better decisions.

1.1 MOTIVATION

During the last few decades, vehicle population in the world has been growing exponentially and dramatically. For instance, accordingly to [84], the number of vehicles (excluding motorcycles, electric bikes, and rural trucks) in China increased, from 5.5 million in 1990 to 109.4 million in 2012, and will probably to continue to increase.

Severe consequences come with a such number of vehicles in our cities, not only for the locals but unfortunately to every people in the world. The rapid growth of vehicles brought concerns about energy security, urban air pollution, and traffic congestion.

The dependency on transportations constitutes a significant problem because it creates another dependency, and that is the need of oil [84].

Alternative fuels for diesel engines have been studied in [4] to explain the impact on performance and pollutant emissions. The authors tested diesel, Gas to Liquid (GTL), and biodiesel and concluded that: **a)** they all achieved the made tests, **b)** Total Hydrocarbons (THC), Carbon Monoxide (CO), and Nitrogen Oxides (NOx) were reduced due to the alternative fuels, **c)** the smoke opacity was notably lower, and **d)** a large reduction in the number of particles' emissions.

Another research in [79], studies vehicles' pollution of multiple pollutants like Carbon Dioxide (CO2), CO, or NOx. They enumerate several alternatives to reduce emissions such as follows: **a)** eco-driving, **b)** vehicle weight reduction, **c)** vehicle power reduction, **d)** increasing dieselization, **e)** use of hybrid technology, **f)** use of biomass-derived fuels, **g)** use of electric vehicles, **h)** use of hydrogen internal combustion engine vehicles, and **i)** use of hydrogen fuel cell vehicles.

We readily understand that the driving behavior is directly correlated with the vehicle's emissions. The average speed of trips has a high impact on fuel consumption, for instance, as tested in [84], if the average speed falls under 30 Kilometers per Hour (kms/h), the relative fuel consumption increases dramatically.

The fuel use and emissions of vehicles directly depend on the way they are driven, even including the daily driving distance and driving conditions [42]. A better understanding of driving behavior certainly can affect their future behavior, contributing to a safer driving reducing the fuel consumption and the emissions. Authors in [69] use an in-vehicle monitor assistant to help to reduce the fuel consumption and reduction of emissions based on the Vehicle-Specific Power (VSP) methodology. They perform an experimental test, with 20 drivers, to detect emissions impact due to the drivers' behavior. They conclude that an eco-driving behave can significantly reduce fuel consumption and pollutants emissions, as well as towards to a change in the driving pattern like reduction in the number of extreme accelerations and decelerations, and time spent speeding.

Traffic monitoring is not only to provide a better usage of the transportations systems but also to help in reduction of ambient impact generated from the massive amount of vehicles that are constantly present in our cities. The importance of vehicles' pollutants emissions triggered the research on traffic monitoring for emissions reduction [35], [38].

A reliable metric to identify vehicles' emissions is the Vehicle-Specific Power. VSP is derived from speed measurements, acceleration, and grade, where it counts in vehicle kinetic and potential energy, rolling resistance, and aerodynamic drag [35]. The methods

to obtain the necessary data to compute the VSP can be achieved via Updated On-board Diagnostics (OBDII) data bus that can give precise values from the car such as speed, Revolutions per Minute (RPM), or Manifold Absolute Pressure (MAP).

As the RPM and MAP are the main influences of the amount of injected fuel into the engine, they generate an increase of the engine load. Therefore, is widely used for emissions modeling, once this metrics has a great correlation with fuel consumption [28], [35]. To get a fuel use rate, it should be appropriately estimated with other models instead of using models that rely on VSP.

Tests made in [35] demonstrate a comparison between internal variables such as MAP and RPM, as predictors of engine load, and an external variable, as the VSP. The authors conclude that the compared points were highly correlated and concordant.

In the future, emission rate models might be incorporated directly into vehicles providing information relatively on how the driver behavior affects the emissions and also share real-time predictions of emission rates [35]. A useful representative of emissions rate data in real-world conditions acts as an undeniable proof of the negative ambient impact caused by vehicles.

Cities are becoming more and more in Smart Cities, and the population is also increasing not only in numbers but also in other aspects as necessities and consumption. Smart Cities are the future, and thus we must build them the best way we could.

To be possible to live with quality of life, we can use and develop mechanisms to provide that quality to people. New communication solutions let us create different methods to gather metrics, and to store them, and then after analysis to then offer it to citizens.

The goal is to include as much drivers as possible, and it will be the best scenario. When everyone has access to traffic information, it will be possible to plan their way to their destination, leaving behind the stress of traffic jams, possibly avoiding lousy weather or very polluted roads, and thus enjoy a trip nicely and without increasing stress on the driver.

1.2 ATMS

Because of the increase in the traffic demand, we need to wisely plan the roads in our cities. In order to face the traffic congestion we could build new roads, but this is not always the best solution due to economical and environmental concerns. The alternative is to make a more efficient use of the existing infrastructure.

The goal of an Advanced Traffic Management System (ATMS) is to efficiently manage existing transportation resources in response to dynamic traffic conditions [67]. Thus, ATMS aims to:

- increase capacity and operational efficiency;
- improve safety;
- increase traveler control;
- improve public transportation services and operations; and
- reduce environmental and energy impacts.

Of course, we can add more definitions to this entity, once it becomes a complete management system with several methods of implementation as well as multiple data sources.

The ATMS is a crucial subfield of the Intelligent Transportation System (ITS). The ITS seeks to provide a good use of the available infrastructure and to improve the information available to the population.

We follow this approach as a base to our solution, as we try to implement and to improve Advanced Traffic Management System.

1.3 COLLABORATIONS

This dissertation was conducted under the framework of the @CRUiSE project as detailed in Section 1.3.1. The fundamental goal is to integrate road traffic impacts into a single analytical framework as improvement of Advanced Traffic Management System.

1.3.1 @CRUISE

The work developed on this dissertation was developed in integration with @CRUiSE project which is expected to last 30 months. The goal is to integrate road traffic impacts into an analysis capability system as an ATMS. Organized upon the following three points:

- Planning a conceptual method to assign a link-based metric to evaluate traffic externalities distinctly by local context;
- Improve the interoperability between traffic models and new data sources;
- Network operations optimization through decision support system.

The research is conducted in a partnership between the Transportation Technology Research group of the Centre for Mechanical Technology and Automation (TEMA), the Research group on Emissions, Modeling and Climate Change of the Centro de

Estudos do Ambiente e do Mar (CESAM) - both from the University of Aveiro (UA) - and the Institute of Telecommunications (IT) at Aveiro, along with the Institute for Transportation Research and Education (ITRE-NCSU), USA.

The first task is to develop a GIS-based dynamic structure that acquires historical and dynamic data such as Floating Car Data (FCD) and thus integrate on a traffic models library with variables like the air pollution or road conflicts. Emissions and noise models are integrated to be later analyzed in different air quality scenarios.

The second task is to improve new sources of traffic data to enhance network efficiency. That need requires innovative methods to deal with different sources of information in real-time to discover the energy and ambient performance within the road network. Implement several vehicles with dynamic monitoring, as mobile applications and/or Global Positioning System (GPS) tracking, and simultaneously, critical points of the network will be measured for a macroscopic vision. Thus, it will be possible to correlate the FCD data with the road traffic data and provide a segment state.

Finally, the @CRUiSE project primary goal is a prototype with a decision support mechanism for traffic management. The project aims to develop an ATMS through optimization algorithms, artificial intelligence, traffic management tools, into a single integration platform. Other optimizing solutions will be evaluated from a centralized perspective as well as from a decentralized perspective for traffic management, thus assuring reasonable and realistic solutions.

1.4 CONTRIBUTIONS

Some contributions were made during the development of this dissertation. Two contributions for presentations with one paper submitted for acceptance and one presentation at a conference.

For this dissertation were developed and deployed three main contributions. One is a mobile application, the other is a car tracker, and finally a website.

Further details are presented on the following topics.

INForum 2017

The 9th INForum - Informatics Symposium (The original title is in Portugese: *9º INForum - Simpósio de Informática*), organized on different topics of interest, spread out through individual commissions. As a major area in nowadays, the growth of Research & Development (R&D) centers triggered an increase in the offer of graduation and postgraduate courses.

The symposium organized by Department of Electronics Telecommunications and Informatics of the University of Aveiro with the collaboration of Institute of Electronics and Informatics Engineering of Aveiro and the Institute of Telecommunications took place on October 12th and 13th of 2017. This year the Symposium operated integrated with the TECHDAYS Aveiro, on October, 12th - 14th of 2017, which took place at *Parque de Exposições de Aveiro*.

Our contribution to this symposium was the submission of a paper to appear in the proceedings at the 9th INForum - Informatics Symposium. It was accepted for presentation, with six reviews, and Track Chairs and Technical Program Chairs have analyzed the scores and comments of the reviewers.

The publication is part of the *INForum 2017 - Atas do Nono Simpósio de Informática, 12 e 13 de outubro de 2017, Universidade de Aveiro*, pages 39 - 49, with the title "IoT Solutions for Traffic Characterization in Smart Cities".

23rd Seminar of the Mobile Communications Thematic Network

The 23rd Seminar of the Mobile Communications Thematic Network (RTCM) took place on July 18th of 2017 at the Amphitheater of the IT, in Aveiro.

The work developed in this dissertation was selected to be presented, as part of Sensor Networks theme, containing a presentation of the work produced and a small demonstration of the system and its functionalities.

Mobile Application

The solution presented in this dissertation presents a mobile application developed to offer the possibility to collect data using people's smartphones. This method allows increasing the amount of data due to the easiness of the smartphones use.

Car Tracker

Another method to collect data is presented in this dissertation and that is the car tracker. This device is designed to be connected to the car data bus and provide data directly from it. It also offers reliable metrics about vehicle's attitude and positioning.

Car tracker is a prototype built in this dissertation and aims to highly improve the quality of acquired data, as well as their accuracy.

Website

The final contribution is a website. This platform provides methods/tools for data analytics, with the capability to check every collected data and information like the last use or online/offline devices at that moment.

The website offers views to watch current climate state, the weather over time up to 5 days, and also a highs/lows values for the current day.

Besides the data visualization, the website also offers a back-office tool, for administrators to manage devices, trips, users, and so forth. Eventually can be added new types of devices with new sensors embedded, and the administrators can add that by accessing the back-office.

1.5 OBJECTIVES

The @CRUiSE project lightly conducted the objectives in this dissertation. All the goals accomplished were also helpful for the project itself, making it closer to the final implementation.

In this document is described the process to accomplish the tasks, which were performed to achieve the final solution. Therefore, in this section, we will discuss our proposed objectives.

According to @CRUiSE requirements, it was developed a **mobile application** to start. As stated above, the project requirements lead us to a need for developing an FCD. Consequently, our first objective was the mobile application where it is supposed to act as a data logger for crowdsourcing.

The subsequent aim was naturally, to give support to the first one, meaning that the mobile application needs some backup functionalities to work properly. The **backend** provides methods and other features to the several components of this dissertation, including the FCD component.

The backend component needs a set of Application Programming Interfaces (APIs) to handle all the elements which compose the project. Another purposed objective, is the **car tracker**, an hardware-based device to be installed on a vehicle to log data from multiple sensors and via the vehicle's data bus interface.

Our goals outline the multiple data sources devices and a support element. So far so good, but the collected data will start to be considerably high. Thereby, the need of data visualization led to the last goal of developing an **interface** to let the user consult his logged data and thus comparing parameters or driving patterns. With this tool, the user will be able to access, for instance, his traveled paths through a map, as well as many other metrics. At this point, we could define our system as an ATMS due to the aforementioned points.

In order to follow some of the requirements from @CRUiSE, we also included the **weather** as a variable of the system. CESAM will provide the data for this new parameter. Then, the interface will include this metric in the data analysis.

The @CRUiSE project aims into a **link-based** data relation. However, our objectives till then were most focused on a travels' approach. But instead, analyzing the collected data from point A to B in the map, and compare the different metrics, it is possible to examine a single link[1] on the map, and correlate all the data belonging to that same location even if from different devices. This approach can give us a significant step in the question of detecting the lousy spot holes in the city, the highest traffic flows, the time of the day that they occur, and many other statistics.

Our ATMS solution will join all of the collected data from each user and perform an evaluation of the **driver's behavior**. This feature will inform the user about his driving pattern, and it could assign a classification to its driving method. This assignment should be somehow, obtained from reference values, like usual speed for a specific road, the atmospheric conditions at that time, or other factors as driver's age or accidents history.

1.6 DOCUMENT STRUCTURE

This dissertation is composed by the following chapters.

Chapter 1 Presents an introduction to the topic, the motivation, the contributions and collaborations made, and the objectives proposed for this dissertation.

Chapter 2 Presents the state of the art about the subject, and also related solutions and techniques.

Chapter 3 Describes the scenario, and a proposed solution. The system architecture is explained on this chapter as well as the requirements for that same architecture.

Chapter 4 Specifies each component of the system, how and what was built and, furthermore the process and the technologies for the development of the solution.

Chapter 5 This Chapter describes a problem and then an evaluation and its results using the solution proposed in Chapter 3.

Chapter 6 As a final Chapter, it describes some final thoughts about the provided solution as well as some future work.

[1] A link is an Open Street Map (OSM) *Way*, which describes a set of nodes in the map, where it includes the positioning information, and other data like telephones, buildings or traffic lights.

CHAPTER 2

State Of The Art

The increase in road traffic leads to high levels of stress in drivers. As such, there has been an increasing interest in developing intelligent ways to help drivers cope with today's traffic. This necessity can be coped using ***a)*** *crowdsourcing to gather data about drivers, pedestrians and other entities,* ***b)*** *Internet of Things (IoT) platforms to manage the data flow, and* ***c)*** *analytical tools to process the information and notify the users. So they can take advantage of safer roads, a better environment and adopt more sustainable choices. In the following subsections the relevant state of the art is presented.*

2.1 GATHERING DATA FOR TRAFFIC MONITORING

For the specific scenario of traffic monitoring, data can be gathered using distinct methodologies: through dedicated sensors, vehicular networks, radio frequencies, using methods based on crowdsourcing, and so forth. In the following subsections, we will describe each methodology in greater detail.

2.1.1 DEDICATED SENSORS

In order to get sufficient data for traffic monitoring it is necessary a variety of sensors disposed on our focus area. For example Unmanned Aerial Vehicles (UAVs) have been successfully deployed in these types of scenarios [41]. UAVs have in their side the advantage of high mobility and low cost of operation. As disadvantages, UAVs are expensive to buy and maintain, do not have good autonomy and may require an experienced pilot, and they can be perhaps a threat to the people and may exist law

complexities. Until this technology is not mature enough, UAVs are not ready for mass deployment.

Another common example of a dedicated sensor is ZELT depict in Figure 2.1, which consists of an Inductive Loop Detector (ILD) able to count vehicles with great precision. Useful for simple managing of car park, where it counts the number of cars inside the park and informs the drivers if the park is or not full with an accuracy of ±5%. There are three fundamental traffic parameters to measure with ILD, such as speed, counting, and capacity, obtained with a single or double ILDs (for speed detection) [58].

Figure 2.1: ZELT system [75].

The inductive loop is placed in the road, a few centimeters over the asphalt and detects the electromagnetic signature of each vehicle as it passes over it [75].

This kind of sensors are almost invisible, because they are inserted in the road. The installation is very quickly, and just needs a small power source to work (for instance a solar panel).

The system works with all kind of cars, because they have metal in the wheels, in some vehicles, like two-wheel motorized, might be omitted on count. So, the ILD solutions aim to control vehicles volume by counting and measuring speed. UAVs might have a great mobility and coverage area, but they still are not ready for traffic monitoring.

The ILD systems might be useful for parking monitoring, or vehicle counting, but for traffic monitoring might not be the best choice. A more covering option is needed for controlling road traffic. Traffic control requires a solution that can cover as many vehicles as possible, with the capability to collect different types of data. ILDs cannot acquire enough metrics, like accelerations or gyroscope, so it is a few limited option.

When it comes to Wireless Sensor Network (WSN), we consider a network of devices, which can merge a set of different sensors, with particular purposes or outcomes.

In [57], a survey is made of this approach, exploring the scenario of WSNs for traffic lights management. Consequently, the intention is to implement a system which dynamically adapts to the traffic flows.

The main challenges are associated with congestion control, reduction of the average waiting time, prioritization of emergency vehicles, among others.

The Road Traffic Monitoring solution covers many sensing technologies, as well as wireless communications technologies. Figure 2.2 shows an application for an urban traffic management on a crossroad, using multiple sensing devices and a central system to perform decisions.

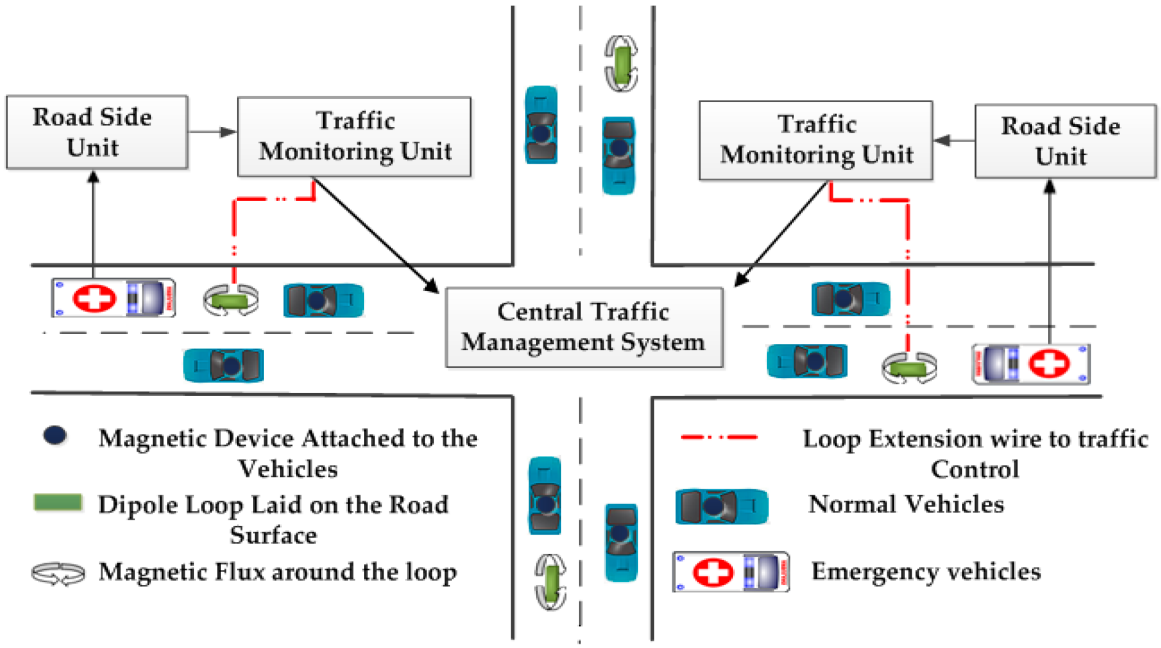


Figure 2.2: WSN in dynamic traffic management [57].

Several sensing technologies can be used in scenarios of WSN. The following list covers some of the possible techniques:

- **ILD:** As in [75], they detect vehicles due to metal in the wheels, by sensing the loop inductance by inducing currents in the object. Offers an accurate count data.
- **RFID:** Uses radio waves to trade data between a reader and a *tag*. Is an economical method to detect vehicles.
- **Microwave radar:** Recognizes regions and captures echoed signals from vehicles by transmitting microwaves. Can measure the speed directly and operate in multiple lanes.

- **Infrared capturing:** Uses infrared energy to recognize regions and capture echoed energy from the vehicles. Such signals must be processed and analyzed to result in vehicle detection.
- **Acoustic detection:** Detects audio produced by vehicular traffic. Can measure speed and operate in multiple lanes.
- **Magnetic perturbations:** Detects presence of vehicles by measuring the perturbation in the Earth's magnetic field. Better than the ILDs in the matter of pavement intrusion.
- **Ultrasonic waves:** Like other wave technologies, this uses ultrasonic waves to detect objects from echoed waves by measuring the time delay between the transmitted and reflected sonic wave. Can operate in multiple lanes, but its performance might be affected by environmental circumstances.
- **Video image processing:** Detects traffic levels using a camera and a workstation for analyzing and understanding the images. Can operate in multiple lanes and offers a great coverage area, but requires specific equipment and some maintenance.

A final note on this survey is that a high number of challenges are implicit on these scenarios. The difficult connectivity and coverage; the problem of the communications; notifications; and alarms.

However, a positive synopsis can be taken from these applications, since a maximum intersection utilization configuration can be achieved. That means that is possible to take maximum advantage of a crossroad, when using this system, by giving priority to more congested lanes and keep the red light on empty lanes, thus reducing the average waiting time significantly and increase the traffic's throughput [57].

ROAD TRAFFIC MONITORING BY SATELLITE

This technique aim to discover congestion problems on roads. It is composed by three parts which are an in-car unit, a server and a mobile satellite communication system (Figure 2.3).

This method works with the in-car system reading the car's position repeatedly using the GPS. Then that location is used to determine the vehicle's speed and the road where it is positioned. The system knows the average speeds practiced for each road under non-congestion conditions, and thus, expect that an average speed to be reached by the vehicle. This information has to be accurate, reliable, timely and complete.

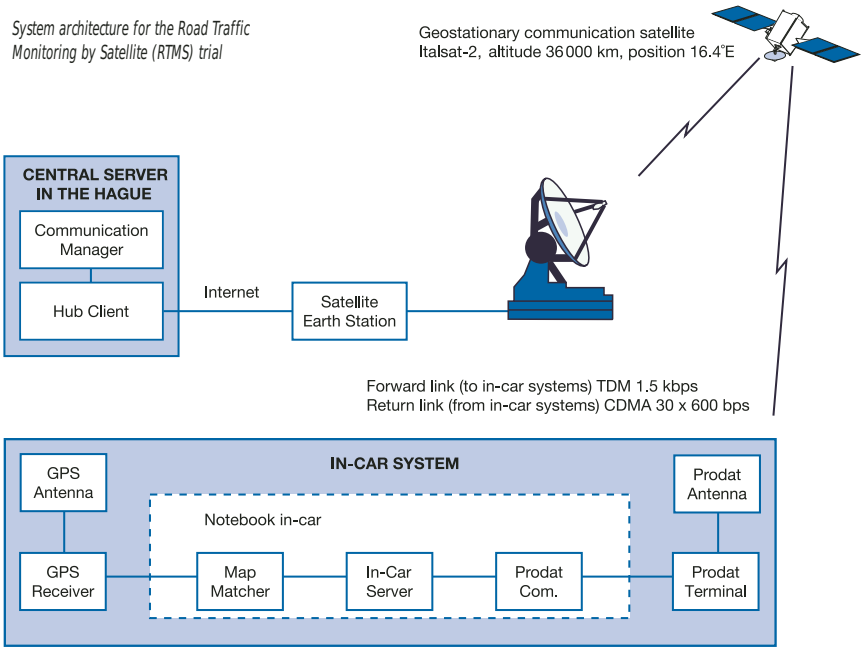


Figure 2.3: Road Traffic Monitoring by Satellite [47].

The traffic information is collected with ILDs, like the one mentioned before, and also with video cameras. This method only gives information from the place where they are installed or in their coverage area which leads to a need for more video cameras to cover bigger areas, and more investment on ILDs.

To get more data is possible to collect it from users, who accept to participate, providing real-time traffic information. This way, it is possible to trace travel times, and other traffic events as incidents, frequent stops, sharp curve, semaphores, or congestion.

Therefore it is possible to automatically detect traffic congestion, analyzing when the car's speed is much lower than the expected speed, with the possibility to refine the algorithm used for traffic detection. A great advantage from this system, is that the server can configure the in-vehicle unit system remotely to adjust routes [47].

This technique has shown to be a great option for traffic monitoring, with an acceptable rate of success. Despite the needs of investment in video cameras, ILDs, and the in-car system, its value of accuracy for each kind of event is satisfactory.

2.1.2 VEHICULAR NETWORKS

Vehicular Ad-hoc Networks (VANET) are part of the Vehicular networks which are very common on ITS systems. These networks aim to advise/warning the driver about an imminent collision, or pedestrian detection, as well as many other applications [62].

The VANETs are also implemented for other kinds of usage like the one suggested in [49], which takes advantage from the VANET by disseminating the information through the network. This approach is implemented by a smart navigation system built to collect data, propagate it to the network, and then evaluate the results.

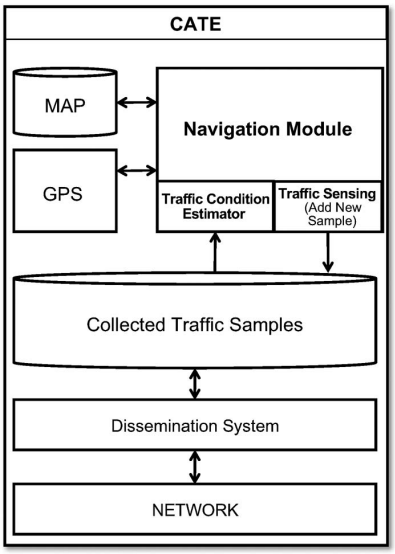


Figure 2.4: CATE's architecture proposal [49].

In [49], a system is proposed and implemented to face the road congestion. The authors developed a solution, depicted in Figure 2.4, named Computer-Assisted Traveling Environment (CATE), a system with three primary objectives: **a)** *Traffic sensing* **b)** *Traffic information dissemination* and **c)** *Traffic estimation*. Each one of the characteristics is well defined and has its purposes. This solution represents a smart system to help in traffic monitoring, in logging traffic data, and to provide processed information to the drivers. The function of each element of the system is the following:

a) ***Traffic sensing*:** Measure metrics related to the traffic as speed, traffic volume, traffic density, and trip time.

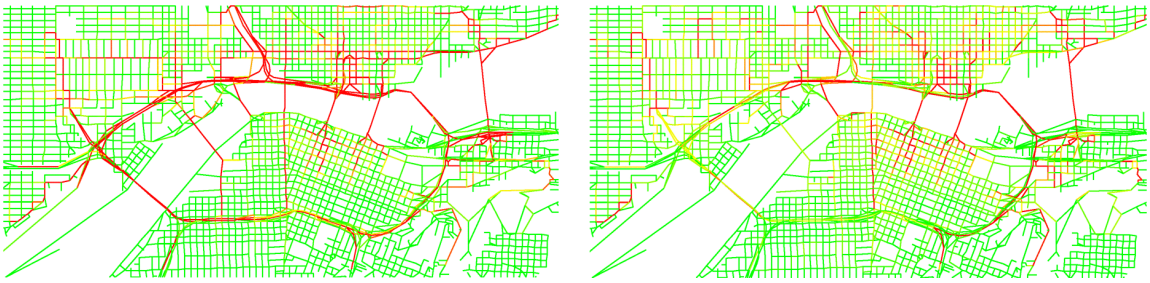
b) ***Traffic information dissemination*:** Vehicles will exchange the sampled information, via Ad-hoc, using the VANET.

c) ***Traffic estimation*:** Each vehicle should be able to evaluate the traffic conditions based on the traffic samples received through the network.

Figure 2.5 depicts maps showing the speed distribution around the city, where the lines in red denote high traffic congestion, the yellow denotes moderate traffic congestion, and finally, the lines in green denote low traffic congestion.

Sub-figure 2.5a, depicts that there are more red links, consequently more congested streets, and low speeds, but on the other hand, sub-figure 2.5b depicts a map with less

lines in red, which they became in yellow and green. This effect was triggered because the drivers were diverted from congested streets to others with less traffic due to the CATE's traffic information dissemination.



(a) Without CATE system. **(b)** With CATE system.

Figure 2.5: Traffic distribution for the average speed [49].

2.1.3 RADIO FREQUENCY EMANATIONS

Vehicles, smartphones, laptops and other electronic devices transmits information in the form of radio frequency. These emanations can be used to estimate traffic density and, given the existence of multiple probes, track citizen movement.

Detecting and capturing electromagnetic emanations might be a difficult work, since we have to deal with different types of waves and modes, that can be originated by direct emissions, like from one device to another, or indirect emanations, created by the interferences between the direct emanations and active electronic components, which induce new types of radiation [78].

SIGNAL ACQUISITION WITH SOFTWARE DEFINED RADIOS

It is possible to define different techniques to discover electromagnetic emanations. One is the emanations that are produced by electronic activity in vehicles, such as their radios, computers, or Bluetooth interfaces. Another technique is discovering the emissions produced by the passengers traveling inside the vehicle. These two techniques are roughly based on spectral analysis on wide-band receivers [78]. The spectral analyzer aims to detect signal carriers, but works only when the duration of the carrier is significant. The wide-band receiver works with a receiver tuned to a specific frequency, then demodulates to its Amplitude Modulation (AM) or Frequency Modulation (FM) and then are used filters to improve Signal-to-Noise Ratio (SNR). This kind of device, the wide-band receiver, is considered to be expensive in this context

and rely on secret requirements [78]. Recently, advances in open source software and hardware, enabled low cost analysis of signals through the use of Software Defined Radio (SDR).

These methods are not perfect and may have electromagnetic emanations still undetected. The wide-band receiver needs a lot of time to cover the whole frequency range, leading to loss of data, and the demodulate process may hide important data.

Is suggested a different method, with the help of an oscilloscope and computing the Short Time Fourier Transform (STFT), to produce a 3D signal, as shown in Figure 2.6, with time, frequency and amplitude [78]. This technique uses a wide-band antenna connected to an Analog-to-Digital Converter (ADC), and with the derived raw signal from the oscilloscope, compute the STFT, without any demodulation. It is yet possible to have a wider spectrum available, detecting frequencies higher than the bandwidth, by removing the anti-aliasing filters.

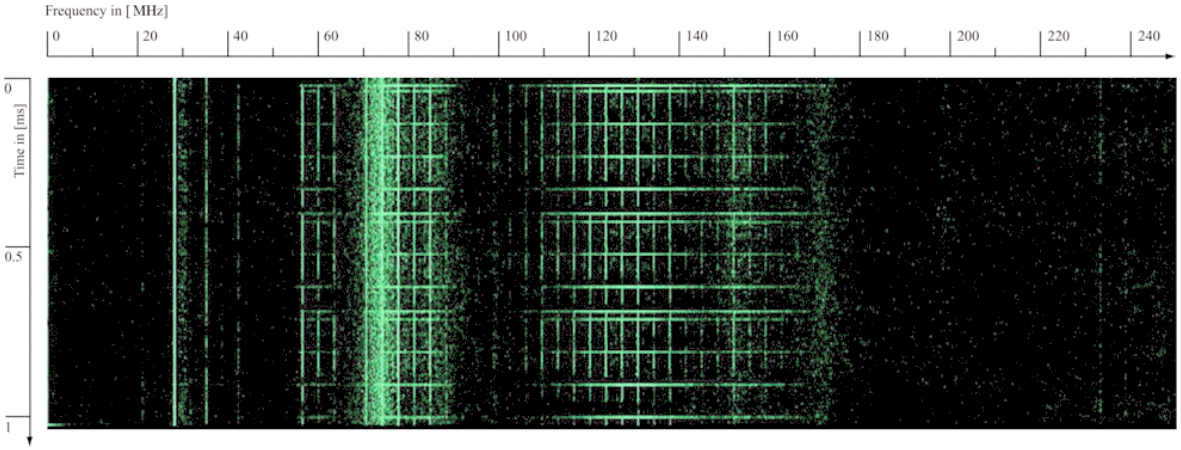


Figure 2.6: STFT of electromagnetic emanation from a keyboard when the key E is pressed [78].

SDR was thought to be a promising solution for interoperability, global seamless connectivity, multi-standard, and multi-mode issues. With the increased sophistication of the wireless devices, the current cellular phones have multi-functionalities such as internet access, cameras, GPS, games, and so forth, which introduced Cognitive Radio (CR) technology [5]. SDR technology implements radio functionalities in software instead of hardware. This results on a higher degree of flexibility and reconfigurability, including the capability to change the channel assignments, improving the spectrum utilization and communication efficiency [8].

The radiometric identification or Radio Frequency (RF) fingerprinting refers to the classification approaches of RF signals [10]. The technology introduced on the research in [10] gives an approach, called Passive RAdiometric Device Identification System (PARADIS), which quantifies radiometric identity of a transmitter by comparing

an observed signal to the ideal in the modulation domain. The evaluation performed by the research demonstrate that the PARADIS is able to accurately distinguish more than one hundred and thirty identically manufactured 802.11 Network Interface Cards (NIC) from a total of one hundred and thirty eight.

The end result is that it becomes possible to fingerprint cellphones, laptops and thus estimate user or car density.

Figure 2.7 shows a summary of the most relevant related work, studied by the research in [10], for identically purposes.

Technique	Type	Goal	Identity Model	Evaluation scale
Kohno et al. [19]	Software meas.	hardware id	clock skew variation	n/a
Franklin et al. [11]	Software meas.	device driver id	compliance with 802.11 standard	17 802.11 NICs
Faria et al. [10]	RF fingerprinting	location distinction	signal power attenuation	135 locations
Patwari et al. [26]	RF fingerprinting	location distinction	multipath channel response	44 locations
Hall [16]	RF fingerprinting	radiometric id	transient properties	30 802.11 NICs
Gerges et al. [12]	RF fingerprinting	radiometric id	waveform accuracy	16 Ethernet NICs
PARADIS	*RF fingerprinting*	*radiometric id*	*modulation accuracy*	*138 802.11 NICs*

Figure 2.7: Comparison of PARADIS with related ideas of the research in [10].

Simple analysis of Radio Frequency signals, even without decoding, allow to estimate the number of users in a given area. That is, by listening to the emanations produced by vehicles (e.g, spark plugs, on board computers, cell phones) it is possible to improve other traffic estimation mechanisms with more accurate data. This way, traffic planning can take in consideration both vehicle movement and density, but also, citizen density [23]. Cell phones are not continuously transmitting but, when moving in a vehicle, they are constantly evaluating their attachment point (Base Station). This causes an increase in emissions of RF signals, which can be captured in the dedicated uplink bands. In the case of Portugal, depicted in Figure 2.8, the uplink frequencies in use are well known, and within the range of SDR equipment and can be used for this purpose.

Decoding of Global System for Mobile Communications (GSM) data with SDR is a process under development, with many advances in the recent years. Is mostly focuses on the downlink channels, and also in call data [11], preliminary work at the GnuRadio GSM (GR-GSM)[1] project show successful decoding of both downlink and uplink channels. This furthers improves the data obtained from monitoring stations, enabling more precise tracking of citizen density, as well as movement in a wide area.

[1]GR-GSM is an open source project dedicated to enabling GnuRadio to decode GSM signals and available at `https://github.com/ptrkrysik/gr-gsm`.

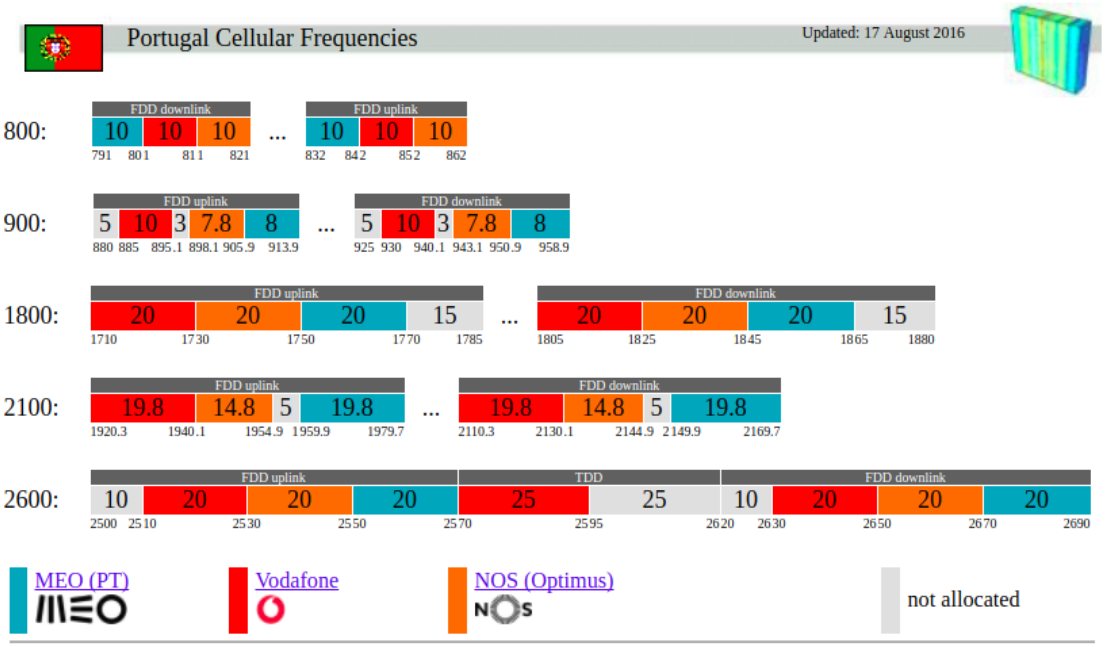


Figure 2.8: Spectrum allocated for GSM/UMTS and LTE in Portugal [71].

2.1.4 METHODS BASED ON CROWDSOURCING

Continuous monitoring of human behavior has plenty of potential to optimize the society. Trajectories and density of pedestrian crowds in a public building, like a shopping mall, can be very valuable for marketing, crowd control, and intelligent energy management. A promising application of these techniques can be applied to traffic monitoring, by controlling and optimizing the routes that are recommended to the users to reduce fuel consumption and carbon emissions on traffic jams [34].

Methods based on crowdsourcing relies on dividing the monitoring needs by a group of people, typically the drivers itself. Each element gathers localized information that can then be processed to obtain a global idea. For instance, the data loggers who are not more than a device that stores sensor data internally. The data can be gathered from multiple sensors like GPS and accelerometer, then stored on data logger system. Nowadays, a smartphone can be a mobile data logger, but most of instruments manufacturers considers a data logger as a stand alone device. The great problem with data loggers is the high price, when most of the time people with smartphones can collect similar data. On the following subsection we detail other relevant solutions based on crowdsourcing methods.

CROWDSOURCING USING MOBILE PHONES

Smartphones are no longer devices just for make calls and send text messages, now they are powerful sensing tools, which allow us to trace individual profiles. This enables a totally different way of measure and classify human behavior and people density in real world.

Nowadays, as the number of smartphones is rapidly increasing it is safe to assume that virtually everyone has a smartphone. The growth of social networks, mobile applications, games, and so forth, boosted the pack of different sensors present on smartphones such as GPS, accelerometers, compasses, gyroscopes, barometers, cameras, microphones, light/proximity sensors, and so forth.

The presence of smartphones in cars, provides a way to implement sensor networks and driver assistance systems, as well as other ITS applications [25]. It is possible to detect road anomalies by evaluating driver behaviors. Analyzing smartphone inertial sensors to calculate the angle swerving and gyroscope drift, can become reliably to detect swerve [70]. There are a considerable amount of research regarding driver behavior analysis based on mobile phones [39], [60], [77].

An example of this work that used call and location data straight from telecom providers to compose dynamic maps with user density and movement patterns is addressed in [23], [36], [80]. Figure 2.9 depicts data obtained by crowdsourcing methods in Aveiro region under the scope of an IoT Platform achieved by [3].

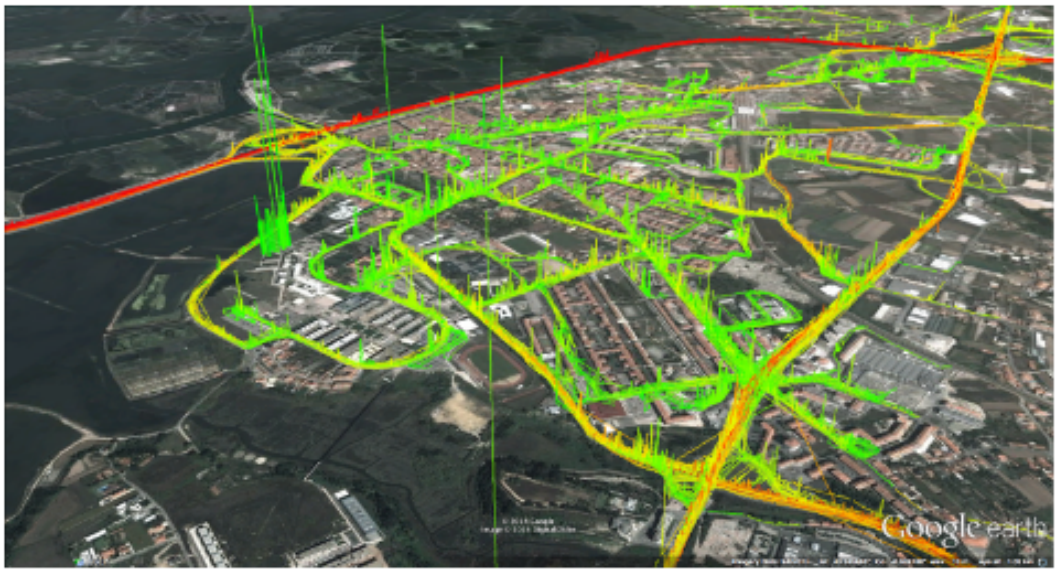


Figure 2.9: Data obtained by crowdsourcing data in the Aveiro region [3], where point height represents vertical acceleration.

For better results in data analytics, traffic research counts on large scale of data collection. FCD can provide big data, once it obtains data from mobile equipments like

smartphones. Thus, a large amount of data is achieved without depending on third party infrastructures [68].

The increasing influence of mobile devices in people's lifestyle, contribute to a more complex variety of data gathering, offering a vast set of different sensors. Recent mobile devices come equipped with GPS receivers, microphones for audio information, cameras for video acquisition, a vast set of sensors, and already with methods for data transmission like Wi-Fi, Bluetooth, and 3G/Long Term Evolution (LTE) connections [14].

2.2 METHODS FOR DATA ACQUISITION

For traffic monitoring and planning is required several data acquisition technologies in order to provide enough data to perform a reliable analysis. Therefore robust methods for predicting, visualize, and modeling are needed to contribute to the efficient use of existent infrastructures, the reduction of emissions and pollutants, and increase the public transportations use [68].

Several approaches performed by researchers include techniques as mapping the streets, use of ILDs, radar sensors or cameras. However, they are not suitable to cover all situations or the massive number of cars in the streets.

Logging data is an old technique but still used nowadays, once it allows offline devices to log data locally. Many applications follow this approach, such as web servers to log their accesses [14]. In some cases is used internal databases to log data offline. Log files provide a method for debugging purposes but can also act as a backup for data acquisition.

Sensing methods are commonly used nowadays, using different types of sensors to measure several metrics. The measured data can then be sent to some platform and stored for further process or analysis [14].

2.2.1 DATA SOURCES

IoT solutions can provide numerous data sources for data acquisition. WSNs and IoT have the property of sparseness which allows capturing all required information without losing data. With the sensors spread over the use case scenario, is possible to cover every need with low-cost equipment and without loss of information [50].

In an IoT scenario several methods can be used for data acquisition like the following:

- Computers
- RFID tags
- Trackers
- Sensors
- Mobile phones
- Cameras

These are some possible data sources capable of providing any required data from measurements or occurred events. Studies in [50] introduce the perspective of data-compression, robust transmission, reduction of energy consumption.

Data sources can provide a context-aware solution, which provides a critical role in deciding what data needs to be processed or not. It allows keeping context information associated with sensors' data, where such awareness offers a straightforward interpretation of the sensor, helping to perform Machine-to-Machine (M2M) communication. Thus, a context is any information related to some identity (sensors for instance) that can be helpful for the interaction between two entities [63].

2.2.2 SENSING

The growth of Smart Cities results in getting something like 50 million devices connected to the internet by 2020 [64]. Sensors are more and more common on everyday objects, and a vast collection of sensors are available for use in IoT scenarios.

To ITS monitors the traffic, it needs to analyze drivers behavior, and identify dangerous situations/events that might influence traffic flow [40].

One type of sensing technology is the fixed sensors placed on the road's pavement as aforementioned in Section 2.1.1. An innovative way presented in [83] allows the detection of vehicles using piezoelectric technology as Weigh-in-motion (WIM) to overcome the limitations of static weighing scales, providing a method to identify the axle load with an accuracy of $\pm 80\%$.

Sensing solutions can be applied by using mobile phones and wearable sensors, like a vital jacket, with significant computing capacity, furthermore with all the embedded sensors they can generate a considerable amount of data [82]. Another solution as in [13] presents a network of smart cameras for urban traffic monitoring such vehicle counting and classification, average speed, queues, and so forth.

2.3 PROTOCOLS FOR DATA REPORTING

Once the IoT devices have limited resources in terms of computation, communication, and battery life, the data transport services should be simple, scalable, robust, efficient in

making near-optimal use of resources, easy to maintain and deploy and also customisable to the need of the applications.

Gathering data from devices is part of a solution for detecting the driving patterns. The storing process is also an essential key of the solution in order to improve the process of data analytics.

Since the majority of devices are battery-powered, with unreliable, intermittent, and low bandwidth connections, special care is needed when discussing data reporting protocols [15]. This Section aims to give an approach to standard data reporting protocols for IoT devices.

The grow of different wireless and cellular networks like Wireless Local Area Network (WLAN), Wireless Personal Area Network (WPAN), LTE, and so forth, network operators want to design their networks efficiently. The interworking between different networks has been developed and researched by standard organizations as Institute of Electrical and Electronics Engineers (IEEE) 802, 3rd Generation Partnership Project (3GPP), and Internet Engineering Task Force (IETF). These organizations desire is to build a standard of heterogeneous network interworking [61].

IEEE defines the standards for Local Area Network (LAN) and Metropolitan Area Network (MAN), targeting the two lowest layers of the Open Systems Interconnection (OSI) model. OSI model structures a network framework to implement protocols in seven different layers. Thus, several protocol standards are defined to allow the interconnection of numerous applications from distinct vendors and manufacturers.

IEEE 802.11, well known as Wi-Fi, has become the dominant standard for WLANs and one of the most deployed technologies. This technology comes integrated into our laptops and smartphones, allowing internet connection without the requirement of being physically wired. IoT devices are frequently designed to be on the move, so a wired constraint prevents many solutions to be deployed unless it uses persistent data, which leads to a need for higher internal storage capacity, and excluding the real-time data feature.

For IoT solutions, systems as Zigbee, Low power Wireless Personal Area Network (6LoWPAN), or Bluetooth have been used for data communications in M2M solutions [1]. LAN/MAN Standards Commitee (LMSC) standardized Bluetooth as IEEE 802.15.1, due to its evolution on the market for short-range communications between constrained devices (especially for mobile phones).

Despite both Wi-Fi and Bluetooth define strong conditions to provide a reliable and easy to deploy solution, the requirements of an IoT solution, with dispersed and constrained devices, make them not a suitable approach. Both were not designed for constrained devices, so they were not intended to provide efficient means for communication and with low-power consumption.

Following subsections present some of the most common protocols for data reporting under the scope of IoT solutions. Also, consider the work from 3GPP in the form of the General Packet Radio Service (GPRS)/3G and 4G cellular technologies as a suitable choice.

2.3.1 IEEE 802.15.4

IEEE 802.15.4 standard for Low Rate WPANs (LR-WPAN) defines a set of specifications recommended for WSNs. Enhancing reduction of power consumption, messages reliability, and decrease of end-to-end delays with specific physical and Medium Access Control (MAC) layers [44].

IEEE 802.15.4 main objectives are manly the energy efficiency, improving channel utilization/throughput, with a smaller payload and more straightforward modulation, improved packet ratio/reliability, reducing collision probability, and delays [44]. Improvements that allow build devices which operate using batteries that last for months or years.

Energy efficiency is one of the leading concern for IoT networks. Data aggregation designs and multipoint-to-point transmission are used to manage power consumption of IoT nodes efficiently. Communication tasks between nodes represent the most significant amount of energy used [2].

The IEEE 802.15.4 standard is the foundation for specifications as Zigbee or Bluetooth. They adopt different approaches and define their upper layers in the OSI model unless the physical and MAC layers [20].

2.3.2 6LOWPAN

As aforementioned, IEEE 802.15.4 works on the two lowest layers of the OSI model, so other standards emerge to obtain a full communication stack. 6LoWPAN is an application of Internet Protocol version 6 (IPv6) protocol designed for low-power wireless networks [12]. Due to transmitting smaller packets, the packet loss ratio lowers and stability data sharing is improved.

IETF standard for 6LoWPAN specifies a mechanism for Low-Power and Lossy Networks (LLN) which aims at saving energy, once it is a protocol designed for constrained devices. Is an adaptation layer for IPv6 efficient packet transportation, with an Internet Protocol (IP) level routing protocol suitable for the needs of IoT mesh networking [33].

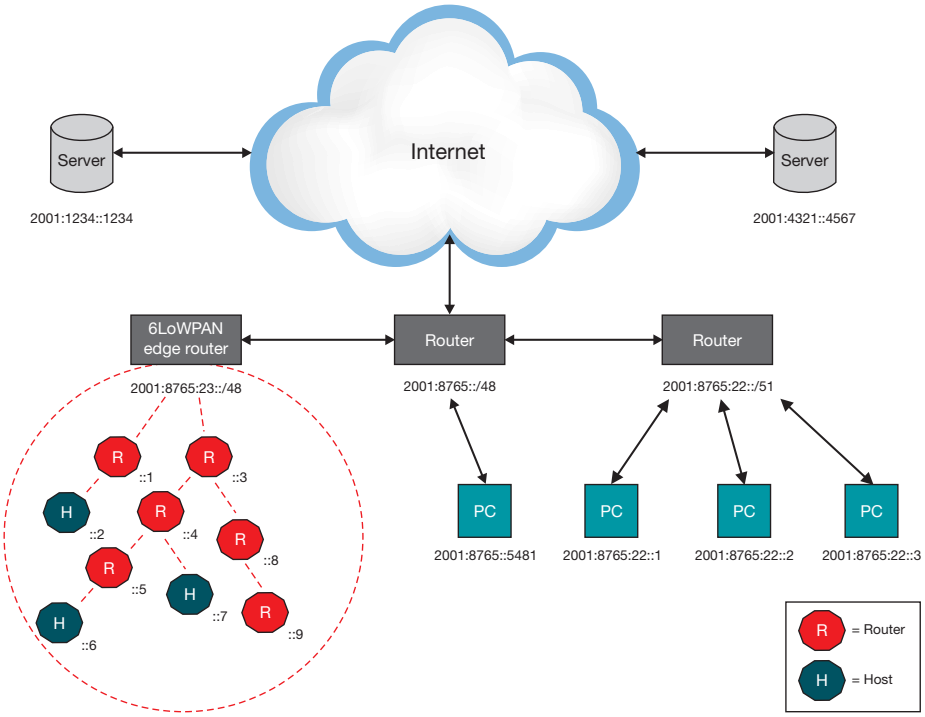


Figure 2.10: IPv6 network with a 6LoWPAN mesh network [59].

Figure 2.10 depicts a scenario of 6LoWPAN use. The main advantage is the capability to support IP-based networks, thus connecting to other networks like the internet using merely edge IP routers. By using the IPv6 protocol acquires several improvements to the old Internet Protocol version 4 (IPv4), such as Quality of Service (QoS), mobility, and multicast, avoiding inherit some of the IPv4 weaknesses [72]. The mean number of devices, when in mesh networks, is far numerous than the number of laptops, for instance, so a significant advantage from using the IPv6 protocol is undoubtedly a much broader addressing space, allowing the increase of small devices with internet access and the creation of mesh networks [59].

2.3.3 BLUETOOTH

The Bluetooth belong to a group of available wireless technologies, such as Near Field Communication (NFC), Radio-Frequency IDentification (RFID), ZigBee or WiFi. The function of a Bluetooth probe device is collect information from vehicle presence and vehicle trajectory information via Bluetooth device discovery.

The main advantage of this technology is its great reliability, with low cost of equipment and very commonly used. Making it a good choice when it is necessary wireless technology in a short area.

As we know the majority of consumer electronic devices today come with Bluetooth wireless capability. So the utility of this technology is the possibility of using a simple antenna, to identify different Bluetooth devices, estimating traffic density, or even speed and travel time [7]. When combined with ILDs, allows the detection of vehicles. The tested scenario in the article [7], which includes a practical application of fusing of traffic monitoring using Bluetooth, GPS and ILDs, concludes that the traffic monitoring with Bluetooth is too infrequent, and showed up to be the weaker point in the test, likely due to its low sampling rate and preliminary technology.

Analyzing another scenario in [76], using Bluetooth scanners, but this time including the duration data as a parameter, which represents the time spent by Bluetooth devices to pass through the detection range of scanners. The detection of Bluetooth emanations may appear from vehicles, pedestrians, bicycles or other atypical vehicles. So the data filtering is an important point, and needs to be carefully dealt to reduce scattering data.

The Bluetooth duration refers to the time between the first detection by a scanner until the time the Bluetooth device falls outside the scanner's coverage area. As depicts Figure 2.11, if the scanner is installed on an intersection, we should expect a duration equivalent to the time occupancy around the intersection, indicating this way, the intersection performance.

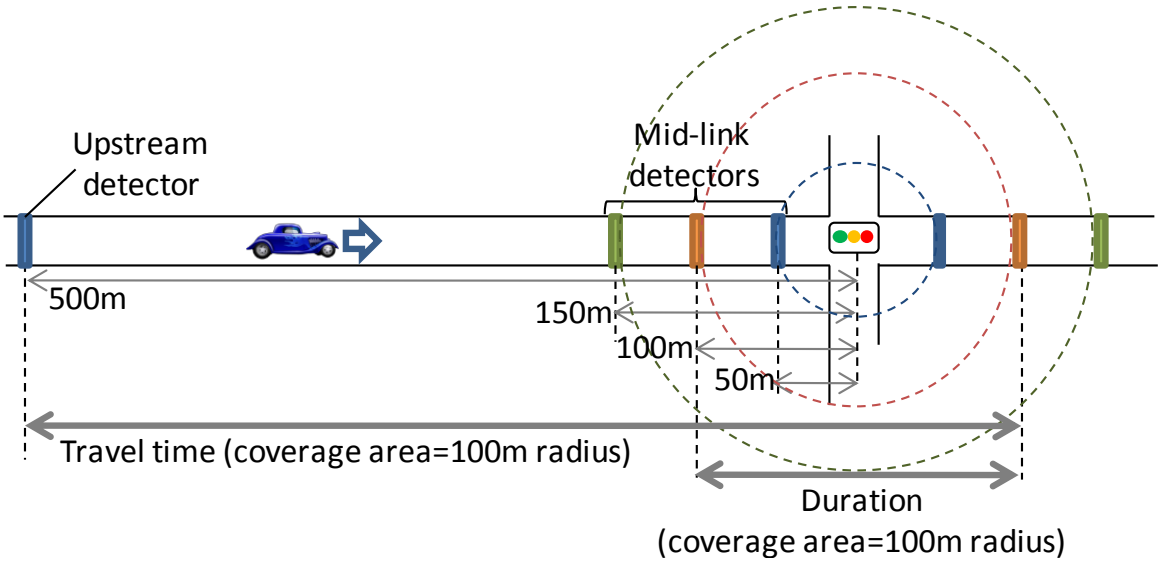


Figure 2.11: Disposal of detectors for detect traffic condition [76].

2.3.4 ZIGBEE

ZigBee, developed and maintained since 2002 by ZigBee Alliance, aims to complete missing standard layers, required to enable mesh networks.

ZigBee sits on top of the two lowest layers of IEEE 802.15.4 standard [33] (considering the OSI model), the physical and data link layers (MAC). Thus, ZigBee protocol defines the rest of the layers, where those layers are composed by: **a)** Application Support Sublayer (APS) **b)** ZigBee Device Object (ZDO) **c)** ZigBee Cluster Library (ZCL) and **d)** Application Framework [33]. A key asset of ZigBee is the layer **c)** which is a library of interface specifications, to provide a repository of commands to be used in application profiles defined by developers.

A missing feature on IEEE 802.15.4 is the network layer for multihop routing of data packets in a mesh network, added by ZigBee implementation of the network layer. Thus, defining short-address allocation, network layer frame format, packet forwarding, routing support primitives, and routing algorithms. Supporting up to 64000 nodes with a short address of 16-bit [33].

Figure 2.12 depicts a scenario of a mesh network topology example with devices connected to a few router nodes and one coordinator which is connected to the internet using a ZigBee IP border router.

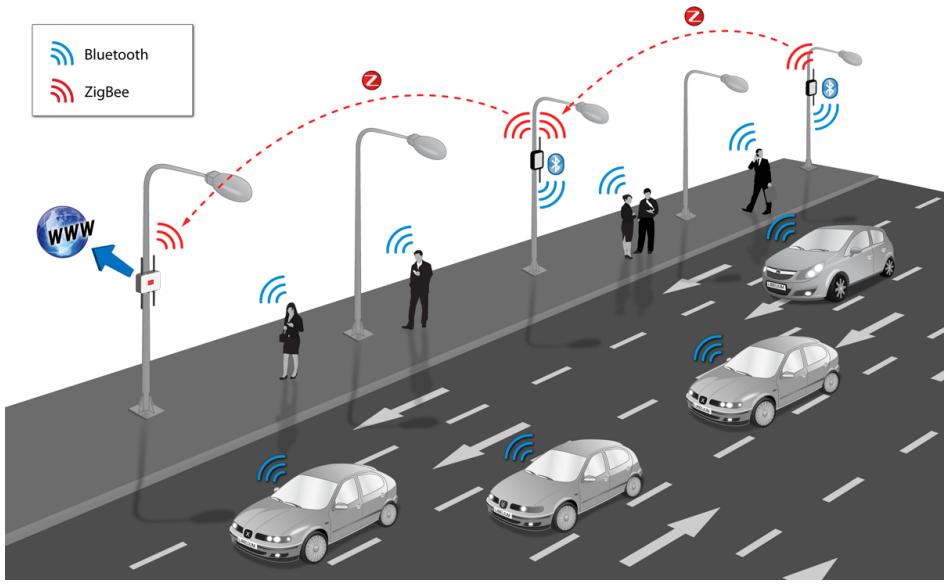


Figure 2.12: ZigBee and Bluetooth in a Smart Cities Environment as depicted by Libelium [52].

As the Bluetooth, ZigBee has also achieved its own market, by being a low-power and low-cost communication system widely deployed in WSNs applications. A variation of ZigBee was introduced, as an open standard, with the principle of use the Transmission Control Protocol (TCP)/IP protocol stack for ZigBee-based solutions. Thus, offering

a scalable architecture based on standard protocols as 6LoWPAN, IPv6, Protocol for Carrying Authentication for Network Access (PANA), Routing Protocol for Low Power and Lossy Networks (RPL), TCP, Transport Layer Security (TLS) and User Datagram Protocol (UDP), to a create cost-effective and energy-efficient wireless mesh network [27].

2.3.5 MQTT

Message Queue Telemetry Transport (MQTT) is a lightweight messaging protocol that relies on publisher/subscriber protocol specially designed for IoT/M2M communications. Handles low bandwidth and high latency communications, supporting simple, small, and low-cost constrained devices working on top of the TCP protocol stack. MQTT is highly recommended to use in constrained devices since it offers low resources usage and it is easy to develop for IoT applications [54].

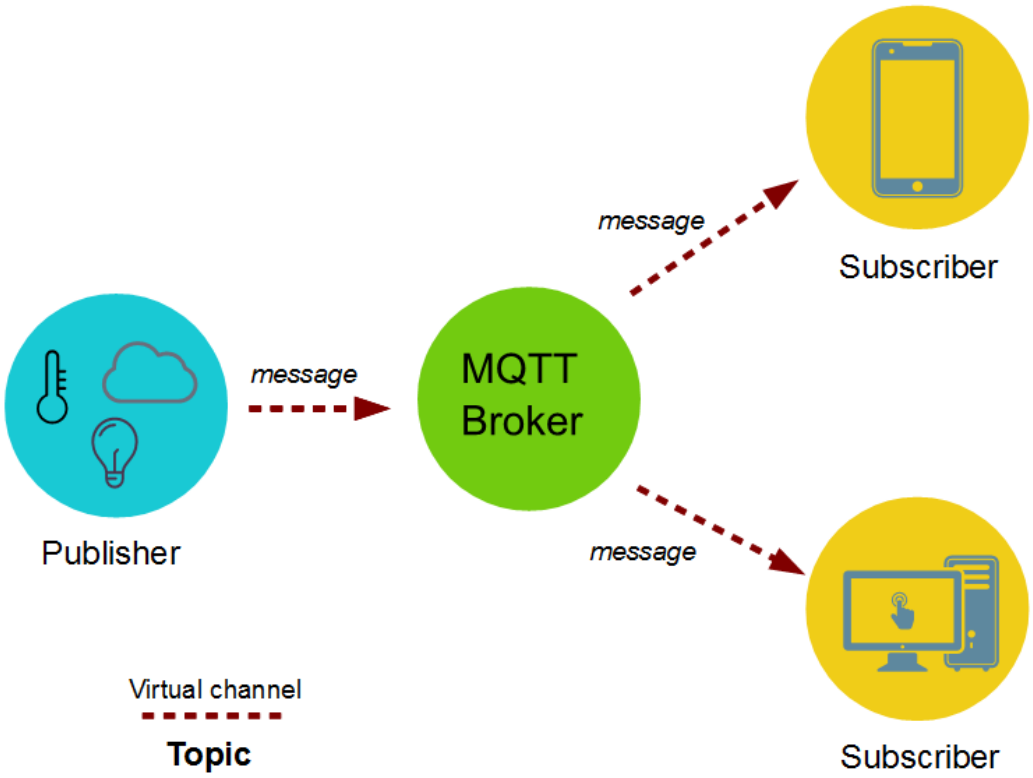


Figure 2.13: MQTT publish/subscribe architecture [6].

Figure 2.13 depicts the publish/subscribe architecture, where a Broker (intermediate server) handles the messages and delivers to subscribers. This model allows any device to publish data to a specific topic and then every topic's subscribers will receive the

published messages. Each topic is a hierarchical structured string used by the Broker to decide which subscriber receive which message. A Broker is the central controller to control the distribution of the data, storing, forwarding, filtering, and prioritizing publish requests from publishers to subscribers [74].

MQTT protocol can optionally provide TLS for confidentiality. Several security methods can be applied such as: no security, pre-shared key, raw public key, or certificates [81].

MQTT defines several levels of QoS for message delivery. Each level demands a different effort from the Broker's behave. Higher QoS levels grant more reliable delivery of the message but have the cost of overloading the network. Brokers keep the message after sending it to the topic's subscribers, allowing the new topic subscribers to get retained messages. Every MQTT clients can set an optional flag to indicate that they are using (or not) a *clean session* connection. Such option, if set to *true*, means that after client disconnect, all of its subscriptions are removed, keeping the client state (the subscriptions and messages with high QoS), so when the client reconnects he receives its subscriptions and most relevant messages. The *Wills* (a message) can be optionally defined when a client connects to a server, specifying a message to be sent to the topic's subscribers when it loses contact with the network [48].

2.3.6 COAP

Constrained Application Protocol (CoAP), defined by IETF, is identical to the Hypertext Transfer Protocol (HTTP) but designed for constrained devices as IoT nodes [27].

The emerging of new smart and mobile devices created a strong interest in domains like energy efficiency and opened a potential interest for CoAP. CoAP is based on a Representational State Transfer (REST) architecture which follows a request/response model made available by APIs endpoints. Servers endpoints provide resources that are available through the GET, POST, PUT and DELETE methods [27]. CoAP adopts the UDP protocol in its transport protocol implementation.

CoAP was designed with constrained devices in mind. It defines retransmissions, confirmable and non-confirmable messages, support for sleepy devices, block transfers, subscription support and resource discovery [59].

2.3.7 HTTP

The HTTP standard protocol is not suited to use in IoT scenarios. It follows a request/response model in a client/server environment over the TCP protocol. Provides communication for IoT devices, however using the HTTP protocol has the problem of long headers and the need of establish TCP/IP sessions for each request-response between client-server [22].

Comparing the HTTP and CoAP protocols, which are very identical, is possible to know that the HTTP protocol is worst in energy consumption and in response time. Therefore, HTTP protocol spends more time in receiving, transmitting and processing packets leading to an increase in the amount of energy used, and also has a longer communication response time [17].

Once HTTP is not suitable for IoT scenarios, Hypertext Transfer Protocol version 2 (HTTP/2) was recently introduced in 2015 by IETF as a successor of HTTP/1.1. Provides a better use of the network and server resources, and reduces the perception of latency. HTTP/2 implements TLS/SSL extension protocol allowing multiple application protocols on the same TCP port [56].

A possible replacing technology is the WebSockets protocol, developed as part of Hypertext Markup Language version 5 (HTML5) initiative. Provide a full-duplex connection between client and server. Clients do not need to continuously poll the servers for obtaining data and removes the need for creating sessions for each request-response, achieving a full push/pull communication scenario between client-server [24].

2.3.8 3GPP LONG TERM EVOLUTION (LTE-4G)

The International Mobile Telecommunications (IMT)-Advanced (LTE-Advanced), or the well known 4G, comes to improve the old 3GPP Long Term Evolution or as we know as 3G [21] (as depicted in Figure 2.14).

This evolution was driven by the requirements of new services for mobile, because of new systems deployed and operated, to accomplish the new regulation of spectrum use [21]. The increase of needs by the new services, and the proliferation of IP has some motivations like most of the services being now over IP, such as Voice Over IP (VoIP). This new use of telecommunications needs more data rate, with no latency, for real-time gaming, web browsing, and so forth. Consequently, with more capacity for total data rate provided on average, which leads to insertion of QoS, to the requirements of the communications nowadays.

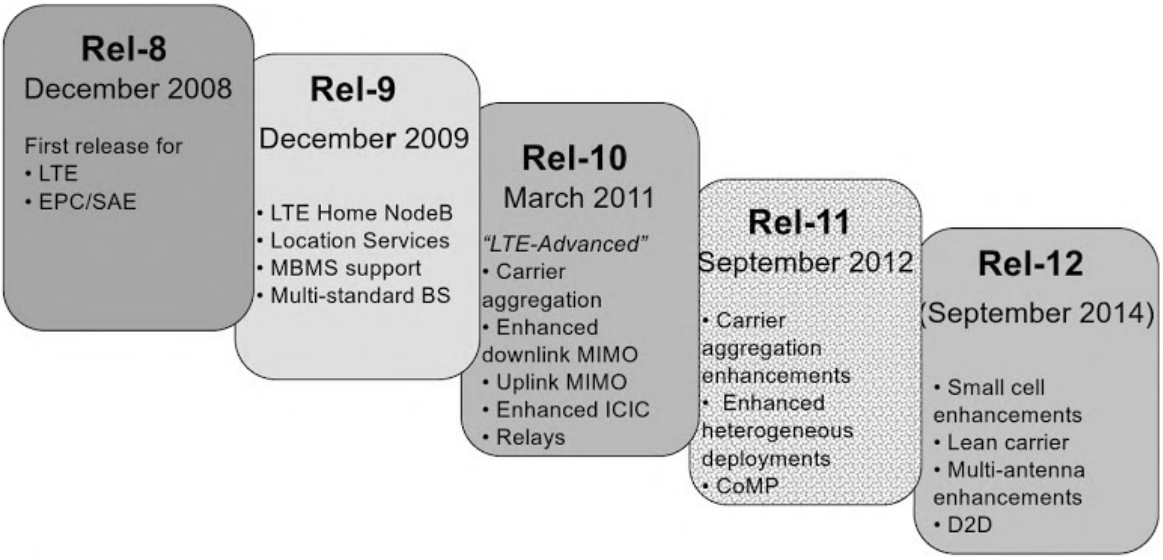


Figure 2.14: List of 3GPP LTE evolution and specifications [21].

This solution is widely available in Europe, and is frequently used as an uplink channel for devices dedicated to remote sensing. The high bandwidth available in the 4G medium enables large scale reporting data, having heavily described and robust interfaces [45]. This is used since a long time, as a technology for uplink of aggregated data, from a gateway to the higher layer platform and services [43]. The constant drawback of 4G is the cost of the modems and the power required for 4G communications, which inhibit the use of autonomous sensors relying solely on batteries or low density energy sources [85].

2.3.9 LORAWAN

Low-Power Wide-Area Network (LoRaWAN), is a wireless technology, from LoRa Alliance, to enable low data rate communications between sensors and actuators for M2M and IoT technologies, for long distances. As we know, the resources, such as power, or processing, on IoT devices is reduced and limited. Long-Range (LoRa) appeared to provide a solution for this issues with a long-range and low-power communication technology. This provides safe bidirectional communication, mobility and localization services. No need of complex local installations to interoperability, and the user or developer gets back the freedom to easily implement the IoT devices [53].

LoRaWAN architecture uses, typically, a star topology, where each gateway acts as a bridge relay for messages between the end-devices and central server. Generally, all endpoints communications are bidirectional, but they also have the possibility of operate in multicast, which allows for instance, upgrade over the air or other multiple

distribution messages, reducing, this way, the on air communication time, since it is limited [53].

Communication between end-devices is spread out on different frequency channels and data rates. The selection of each parameter is a trade-off between range and message duration. Due to the spread spectrum technology, there are no interference between channels [53].

With different elements of configuration on LoRa wireless system, the values that can be reached on an average situation are the following [66]:

- Long range: 15 - 20 km
- Millions of nodes
- Long battery life: up to ten years

The most widely used frequencies/bands are: 868MHz for Europe, 915MHz for North America, and 433 MHz band for Asia. In configurations regarding both Point-to-Point and Point-Multipoint, which is the base model for LoRaWAN, the LoRa gateway acts as an aggregation point for sensors, monitoring the environment, relaying the information for servers and IoT platforms. This typical example is depicted in Figure 2.15.

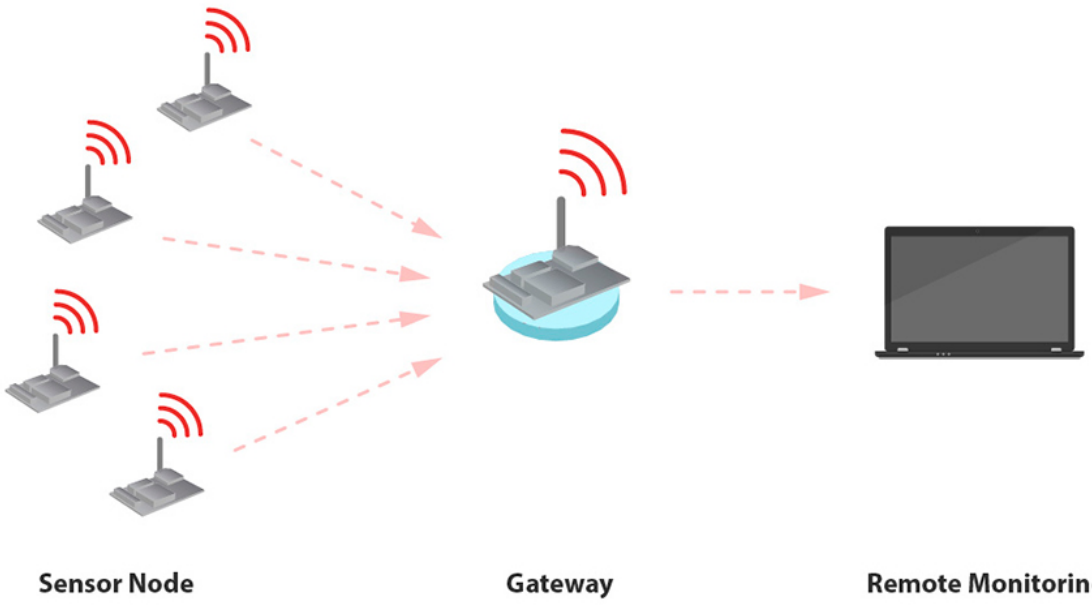


Figure 2.15: LoRa network architecture [51].

A relevant constraint related to the long-range capabilities of LoRa and LoRaWAN, is the inherent maximum duty cycle of the Industrial, Scientific and Medical (ISM) bands where it operates, which can be as low as 1%. Effectively, the long-range characteristics are inversely proportional to the effective throughput, due to the fact that messages sent

for higher distances will occupy more spectrum time, reducing the number of messages that can be legally sent.

CHAPTER 3

Solutions and Scenarios

For an accurate data acquisition and the analysis of the scenario, we must do it in the best possible conditions. Everything must be planned and well defined, therefore in this Section, is described the test scenario and its requirements, the proposed solution and its use cases.

3.1 scope

The ideal scenario would be to have a real prototype built and an FCD to make real-world tests. With the collected information available on the dashboard, the user should be able to compare and analyze it.

The idea is to obtain a system capable of evaluating a driver's behavior, and be suitable to trace individual information about different drivers. Complementing this system, there will be measurements of other parameters in the critical points in the city.

Measure critical points of the city is a significant aspect to include. The primary scenario for us is having multiple data sources collecting data to a platform which then can be accessed and analyzed. As an improvement, the critical points in the city are measured obtaining traffic parameters like link occupancy or traffic flows. With all critical points in the city covered, it allow us to process multiple data sources, as the tracking device, the FCD, and the road measures, to match everything and provide a fulfilling road characterization.

To create a system which will evaluate the driver's behavior, it is necessary to have an infrastructure (hardware and software) prepared to deal with the collected data from the system. Such system will have more than one type of data sources such as car data bus, motion sensors and smartphones as an FCD.

The challenge presented in this dissertation has a focus on data acquisition, despite that fact, we know that nowadays collecting traffic information is not enough to accomplish an efficient traffic management. We also include more than only a solution for data acquisition, in a full-stack implementation.

Since we seek for a complete solution for the scenario, we need to have implemented the following features:

- Several tracking devices for each vehicle plus an OBDII connector for each one;
- The mobile application installed on a Smartphone;
- Backend server;
- A platform to store all collected data;
- Device management system with association to the user;
- Measurements in the critical points of the city relying on fixed probes;
- Dashboard with data analyzing tools;
- Matching of all data sources to provide the state of traffic of the road segments;
- Driver characterization algorithm;
- Route optimization and eco-routes suggestion.

Each vehicle tracking device must be built and deployed on each corresponding vehicle, along with the OBDII adapter. It may be considered intrusive since the OBDII connector may not be easily accessible on every car. However, this device will represent one of the best data sources on the solution.

The most straightforward component is the mobile application. With just a smartphone, the user installs the application and starts to collect information.

For both components is necessary an identity system, to perform an association of the data to the user, as in the dashboard, each user can access only to the data acquired by its device and/or smartphone. Although, the data collected on the critical points of the city is available to all users and is matched where needed with other parameters.

With the critical points in the city covered with measuring probes, the system can provide metrics about specific road state. Since we have all the data sources on the scenario, the system must be able to provide tools to analyze everything with considerable detail and options. Thus, matching different types of data sources will allow the system to make a reliable characterization of each driver, by analyzing the collected data and the fixed probe's measurements. The driver behavior patterns can be detected and provided to the user, letting him know if is or not an aggressive driver.

An eco-routing is a path from one place to another, but the difference of this method is that it gives a path according to the most economical way to reach some destination, while common routing methods provide the quickest route or the closest one. This eco-routing uses our system to map the most fuel-efficient route available.

Such approach counts on traffic and weather conditions, and other metrics to provide a route optimization.

3.2 REQUIREMENTS

As explained in Section 3.1, an infrastructure of hardware and software is needed to have all the system working. Starting with a Floating Car Data where we use a smartphone to do the work, satisfying the first requirement of the project. This requirement is constituted by just a smartphone application.

To store the information collected by the FCD it is required a system working as a backend to handle all the needed support. This backend is mainly a server running an appropriate software acting as a backbone for the FCD as well as for the other data sources like the ones that will be addressed further.

Another item of the project is the tracker. This tracker is composed of sensors, with a system capable of collect information of the car where it is installed and then publish to the network. The hardware needed for this item is the tracker. The complete composition of this item is a hardware, the tracker, and its system.

Like aforementioned, the backend is a crucial part of this architecture. It acts as a source of information providing devices information and it also has the function of providing a dashboard created to show all information relevant to the project.

In order to make all this make sense, it is necessary an IoT platform. This IoT platform is an all different systems, as it consists of a network of services with a database, services, and APIs to have a decentralized system to store data and interconnect different devices [3]. This platform is mentioned with greater detail in section 4.1.

Although very common in IoT solutions, the connection between the system components is a requirement. Several protocols provide lightweight mechanisms to implement IoT devices. On the other hand, other components, like a backend server, are capable of more robust protocols, which handle better communication solutions.

3.3 PROPOSED SOLUTION

The final proposed solution is able to recognize where are the hotspots in the city, traffic jams, traffic slowdowns, and to identify the most problematic zones in a city. To help the drivers avoid these situations of stress, congestion, and pollution exposure, was proposed a solution to support drivers and also to improve the cities growth sustainably, with traffic congestion planning, and efficiently manage the transportation systems.

The goal of this dissertation is to join different data sources and combine the data in one place. The equipments to collect data will be placed in a vehicle, and then the collected data will be sent to a platform, where they will be stored and processed to further analysis on the dashboard. Figure 3.1 depicts an overview of the proposed solution in a small case scenario. In a real scenario, it is expected multiple vehicles using this solution to contribute to traffic management, as well as many other aspects mentioned in the scenario presented in Section 3.1.

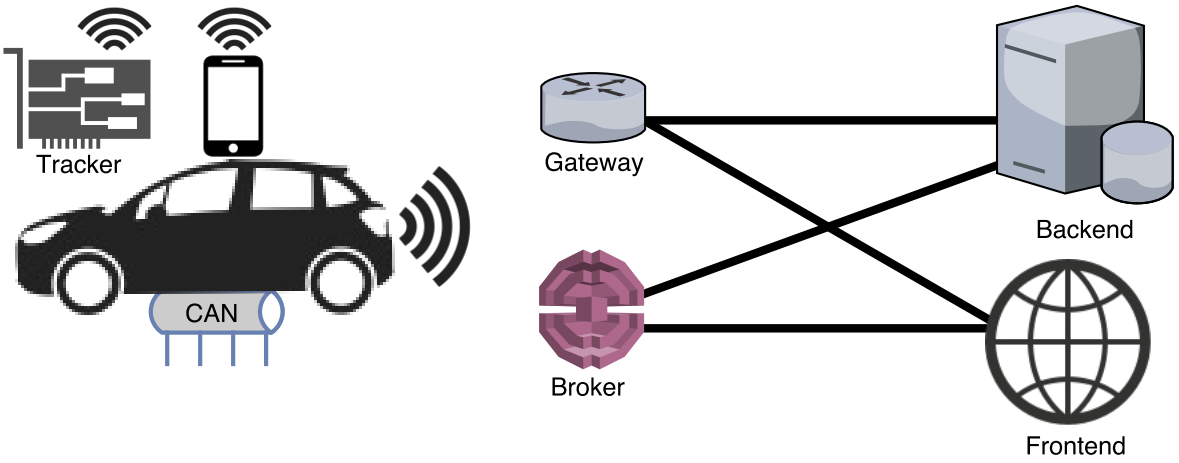


Figure 3.1: Scheme of the proposed solution.

Beginning with data acquisition, it was thought a realistic solution for it. So, emerged the idea of a prototype device to provide a vast dataset to be logged, with high quality and reliability. Once this prototype is created, it should grant the majority of the requirements defined to detect situations of stress, congestion, and even to log only the vehicle's positioning. This solution provides a decentralized option of data source, and it might be the best one according to the scenario.

Despite the prototype be the best option to use, it requires much more work, and it does not cover as many vehicles as we want, due to the difficulty in building a significant number of this kind of devices. So, the solution was to develop a mobile application capable of logging the maximum number of useful parameters, similar to what we can get from the vehicle prototype. Thus, the proposed solution has a boost in covering numerous more vehicles, simplifying the way how the people can get a method to log traffic data.

With the acquired data there is a demand for a platform capable to support large amounts of data. Consequently, it must be accomplished by the solution proposed. The final solution provides a platform capable of storing all the collected data of each device.

The stored data is not useful if there is no way to analyze it. The best way to show virtual information to a user is naturally, through software. Such component

was proposed to be developed and provided by our solution. An element to provide information to the user, always represents a challenge, once we have to produce something to people understand, and not something, for instance to distinct systems communicate. A large number of solutions are available on the internet, but the proposed solution in this dissertation uses its own structure.

Once the infrastructure to support the proposed solution is ready, it is necessary to develop methods and techniques to obtain the true purpose of this work, a system to support the traffic management, and to provide useful information to its users. Combine collected data and weather information to provide statistics from the driver behavior. Moreover, a significant information wanted by every driver is undoubtedly an analysis of its driving behave. This dissertation explains a method to process the acquired data, and then provides this information to the drivers.

SYSTEM ARCHITECTURE

From the proposed solution a system architecture is presented in Figure 3.2. It aims to display the fundamental components belonging to the system, and its prime modules.

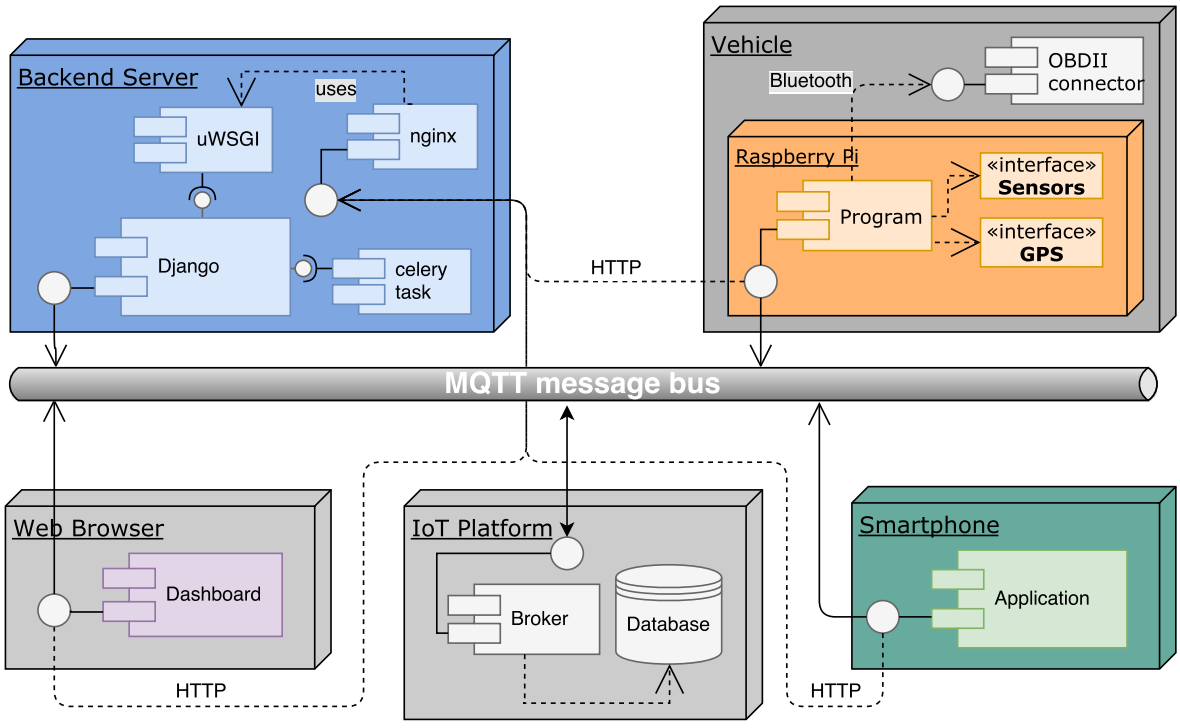


Figure 3.2: Solution's System Architecture.

Web Browser and *IoT Platform* components were not built on with the solution, as they already exist, or they are external tools/requirements which were not necessary to build. In this particular case, the web browser or the vehicle are not our responsibility

to accomplish. However, the IoT Platform component already exists, being supported by the IT and developed under the work on [3]. All other components exhaustively use this platform, either for storing or fetching data and it is detailed in Section 4.1.

Raspberry Pi and *Smartphone* components are the data sources devices. These represent the two types of devices implemented by the solution. Although, the system can support, for instance, more types of data sources, like crowdsourcing or fixed probes. Both devices are detailed in Section 4.3.1, for the mobile application, and Section 4.3.2, for vehicle's tracker.

Raspberry Pi component has multiple sensors embedded to collect data. The GPS module is included in the set of sensors of this device. The device communicates with the OBDII connector which is connected to the vehicle's data bus, providing then a set of parameters highly significant to determine the driver behavior.

Smartphone component acts fundamentally as a mobile application with the purpose of log data from the smartphone's sensors.

Backend Server component is the central piece of the system, manages the devices, as well as the trips, and is also the place where the frontend is served. The complete component design is detailed in Section 4.2. In this component are deployed several systems belonging to the infrastructure. *Django's* element is the most important piece of the server, once it does the devices' and trips' management, serves the website and also handles the weather acquisition using a *Celery Task*.

Other aspects are addressed in Chapter 4, which explains all the process to built the components, the technologies, the methods, and requirements, resultant by the implementation.

3.4 USE CASES

A simplified method to display how the components interconnect between them and to understand the role of the drivers in the solution. The following use cases try to cover the most important actions between the essential components of the system. There are several use cases detailed for each identified actor, and they cover all actions that can be performed by each one of the actors.

Figure 3.3 presents the drivers' use case. The logging components of the system need human interaction to be deployed and executed. Either to start/stop the data acquisition or to install the devices in the vehicle and make the necessary configurations. Before starting to drive, every driver needs to initialize both devices or at least one of them. In this use case, the driver begins by placing the devices in a secure position inside the vehicle and, if needed, makes the necessary setups. For instance, the mobile

application requires the user interaction to optionally setup a trip name, and then starts the data acquisition.

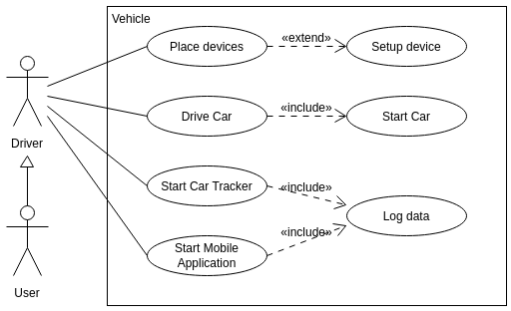


Figure 3.3: Drivers' use case.

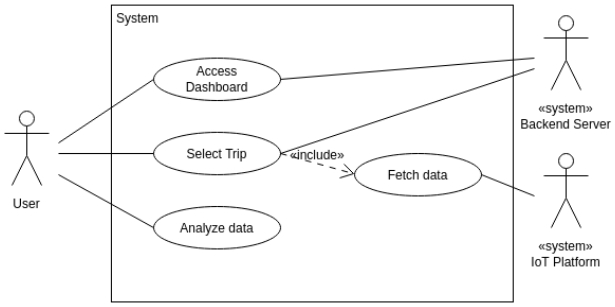


Figure 3.4: Dashboard's use case.

Figure 3.4 presents another use case dependent on human interaction. The system provides tools to analyze and display the collected data from the devices. Users operate that tools and such users are the described as drivers in Figure 3.3. The use case depends on two system actors, the *Backend Server* and the *IoT Platform*, both act as a support system to handle the client requests for data. There are three primary actions that the user can perform, which are:

a) Access to the dashboard through a login system;
b) Select a device and one of its trips;
c) Inspect and analyze the output given by the dashboard.

Actions **a)** and **b)** are handled by the Backend Server, while action **c)** is handled by the IoT Platform, where every collected data is stored. There is a login system to provide access only to registered users, as well as to relate the trips with the driver. All devices and users are unique in the system, to make possible the identification of each one within the system.

Figure 3.5 presents a use case for the proposed devices. Both devices have an identical set of actions. They perform an initial setup before starting to logging data, defining a trip name and set the start time. Once setup finished, they proceed to sensors initialization, connecting with available sensors and existing modules as the OBDII connector or the GPS receiver.

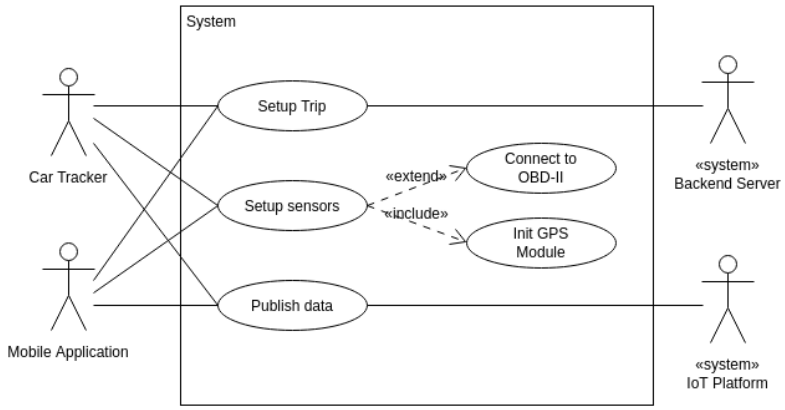


Figure 3.5: Devices' use case.

3.5 COMMUNICATION METHODS

As discussed in 2.3, IoT solutions demand the adoption of lightweight protocols for device communications. The method adopted to implement in the devices should be proportional to the capacity of the device that executes it. Thus, a set of protocols can be found to perform the communication task.

Section 2.3 presents a survey of nowadays most used reporting protocols for IoT solutions. Although many protocols could be capable of accomplishing the communication protocols requirements, only a few protocols were used, therefore it is not necessarily required the implementation of all of them.

The communication between the car tracker and the OBDII adapter is implemented using the Bluetooth protocol. Here the choice was restricted, once the tested adapter supports only this communication protocol.

On the other hand, a different type of communication protocol can be used to connect the car tracker to the vehicle data bus. The Universal Serial Bus (USB) works through cable, and the user can opt to use one of the two connection options, since the system is prepared to handle them.

There is not communication between the devices, thus it is not necessary for this solution. However, the devices communicate with two entities. All the available devices types perform the same actions, despite the different logged parameters, but the process flows exactly in the same way.

As discussed in Section 2.3.5 is the most favorable protocol for publishing the acquired data, which let us to publish without the need to handle message queues or specific destinations to send data. The only requirement is the former creation of a topic specifically for that purpose.

For data reporting and according to the tests, the MQTT protocol showed up to

be the perfect choice to implement on the devices. Despite serving such purpose, it does more than reporting data. It also provides a real-time data reporting method. With this, the system can provide real-time information to the users. For instance, it is possible to show the number of active/inactive devices on the system without refreshing the web page, due to the framework used to build the dashboard (AngularJS). That framework implements the MQTT protocol and subscribes to the topic created for data reporting, and receives then every data that is being currently published to it.

While MQTT protocol handles the data reporting, there are other requirements present on the system, like the trips management. HTTP protocol is directed to a client-server approach, which is where it is most used. All devices must access the Backend Server to set up a trip on the system. For this task is needed a protocol, not for real-time data neither for data reporting, but capable of performing queries to the server APIs. To use the backend API, the devices use the HTTP protocol, once is it the only available protocol by the server.

To provide the collected data to the users, the system must fetch it from the IoT Platform. The IoT Platform provides an API over HTTP, where the system obtains the necessary data, making specific queries to the platform.

CHAPTER 4

Implementation

The idea of collecting data from different sensors and combine them to have some kind of characterization of the traffic, presents serious challenges. In part because sometimes it is not known what data can be useful, or what frequency can be used. More often, data becomes meaningful after analysis and otherwise irrelevant sources of information become vital. As a simple example, it should be considered that the driving behavior changes during the time of the day, the weather conditions, the day of the week, or even due to the music that it is being played in the radio. While we cannot observe all aspects contributing to the behavior of a driver (as some as intrinsic to the driver), our approach is to acquire as many sources as possible, and then the data is correlated and analyzed.

4.1 IOT PLATFORM

The IoT Platform is an external project, it was not part of the solution presented on this dissertation and belongs to a project named Smart Cloud of Things (SCoT) [3]. Nonetheless, it is a key component of this dissertation. The platform represents the connection point between all devices including the backend and dashboard.

The platform is spread out in diverse components, each segment has a specific function and they represent a decentralized system.

SCoT is composed by several pieces, as depicted in Figure 4.1, and needs a core network of machines working together. Certainly, this has both advantages and disadvantages, in one hand, it easily allows the exchange of a broken machine then restoring it using a backup but on the other hand, it becomes complex to manage and setup a system so distributed.

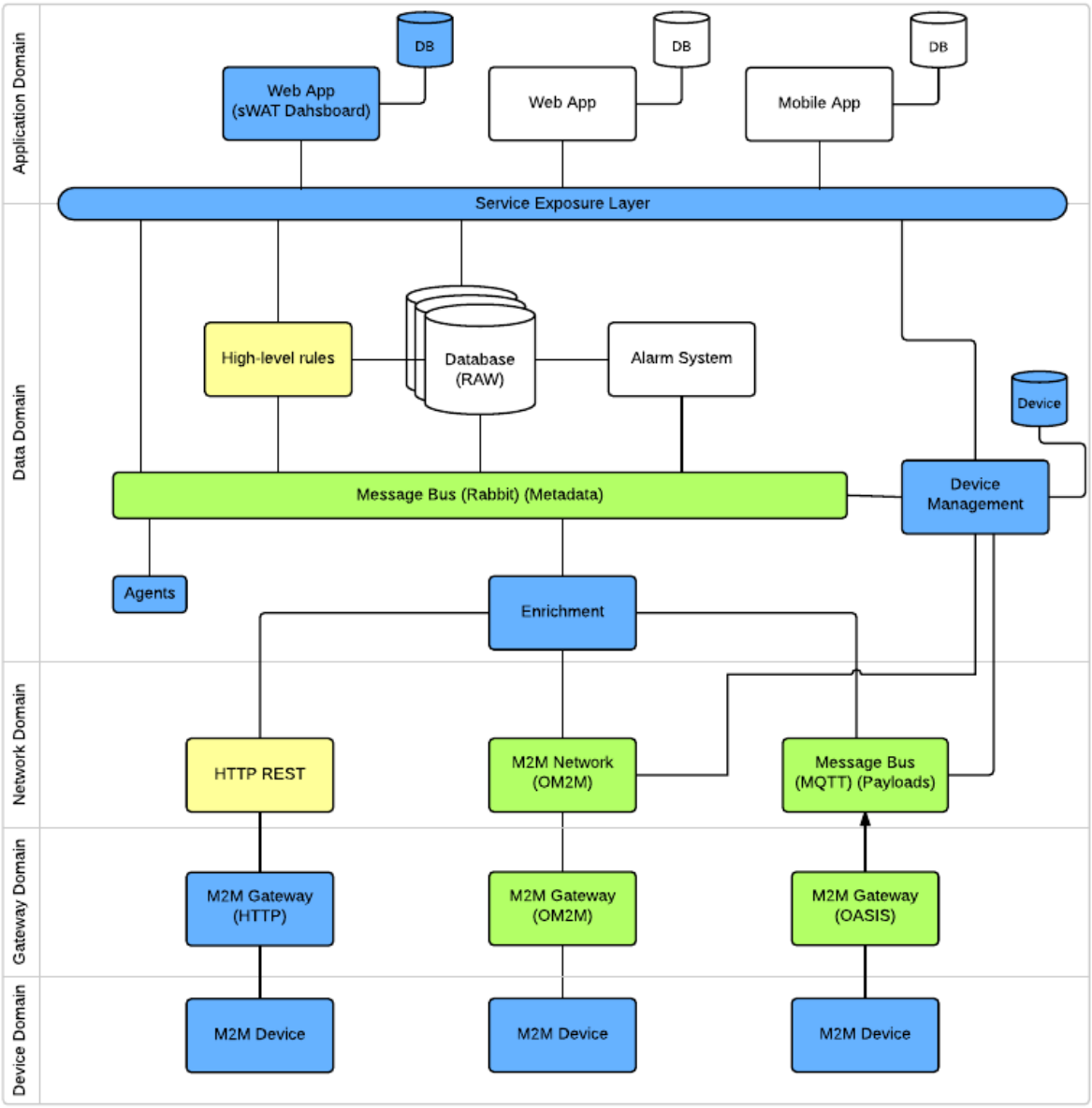


Figure 4.1: SCoT Functional Architecture [3].

This platform allows working with different communication protocols either for HTTP or M2M to be used by the IoT devices, where most of the times, the hardware, and power consumption are limited.

This platform works with three main components, the MQTT protocol, a message bus with Open Source Enterprise Messaging (RabbitMQ), and the storage component using Apache Cassandra database (Cassandra).

All data that passes through the message bus is stored in raw format, to the Cassandra database. To access the stored data, the platform provides an API to make queries by given Uniform Resource Locator (URL). The URL must have required information about the device and what should be searched. There are optional parameters

that can be sent to filter even more the search, for instance, it is possible to get data by name, or between a timestamp.

```
https://io.atnog.org/data/<tenant>/<group>/<product>/<device>/<sensor>?s=15000
&q=["payload"]&f=list&api_key=<device_key>&b=1501369200000&e=1506726000000
```

Code 1: URL to query data in the database.

Code 1 is an example to query the database. Some are required parameters like `api_key`, but the parameter `e` is optional and represents the end timestamp, which is the limit timestamp for the results. With the parameters `b` and `e`, we can limit the search between two timestamps, with a precision of milliseconds.

To the client, this is absolutely transparent and it does not need to work directly with data access. This feature is made by the dashboard which will be explained in Section 4.4.

4.2 BACKEND

To perform all features mentioned in Section 3.1, we developed a backend to support the requirements of the scenario.

The backend is the central point of the project. To develop this component, we choose to use the full-stack Django framework, because it provides many functionalities, which are useful, like relational database models, security, readability and maintainability, and all in one place, highly scalable and robust [73]. As we need an API, a data model and a place to host the dashboard, the choice went to the Django framework. Another motivation was clearly the programming language, once the Django is built in Python and it is a programming language very adopted nowadays. More precisely we used the version 1.10.5 of Django framework and with Python 2.7.12.

Figure 4.2 describes the flow of the server, where the client communicate with the server, and the several layers of software until it reaches the Django framework, each one with different purposes.

The Web Server Gateway Interface (WSGI) is a specification or an interface for the interaction between a server and an application, in this case with Django framework. The uWSGI is a project used to implement Django interface, with the goal of hosting services. These are different services, proxies, process managers and monitors who are all embedded in a single API that can be added as a plugin.

After the Django framework being served by the uWSGI server, is not supposed to let the client communicate directly with the uWSGI. The solution was to deploy

the Nginx, an open source software for web serving, that has the capabilities of reverse proxying, caching, load balancing, and more [29]. The main advantages are the great speed and the capability to be a reverse proxy server. In the Nginx configuration file we can define different locations (for URLs) and ports as well as other functionalities.

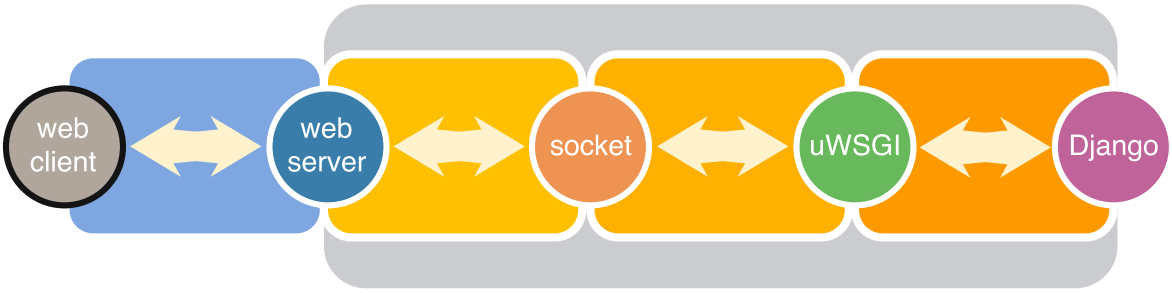


Figure 4.2: Sequence of steps when a client communicates with the server.

In appendix A, it is possible to see how the Nginx was configured to behave correctly when dealing with different requests. This configuration is prepared to work with Unix sockets instead of the TCP port socket. This means that the uWSGI is configured to produce a Unix socket to be accessed by the Nginx.

After this configurations, it is safer and recommended to use a processes monitoring tool. This tool is called Supervisor and its purpose is to control processes on Unix operating systems. With this tool, we are able to configure the Supervisor to start the application on system startup, and if something goes wrong and the process terminates, the Supervisor relaunches the application automatically, if set up for that. Code 2 shows how the Django framework is configured to start with uWSGI.

```
[program:wsgi_cruise]
directory=/home/backend
command=uwsgi --virtualenv /home/venv --uid www-data --gid www-data --socket
↪  /tmp/wsgi_django.sock --buffer-size=32768 --workers=2 --enable-threads --master
↪  --need-app --chmod-socket=664 --wsgi-file dashboard/wsgi.py
stdout_logfile = /var/log/backend_dashboard.log       ; Write log messages
redirect_stderr = true                                ; Save stderr to same log
autostart=true
autorestart=true
;user=www-data
stopsignal=QUIT
```

Code 2: Supervisor configuration of the Django framework with uWSGI.

The Django's project is a set of applications, as depicted in Figure 4.3, where the `settings` represents the main application, and it is where it is defined the main properties and the base routes for the APIs. All client requests go to the `settings`' API, and then the routes are mapped to the other APIs of the project, as the `authentication`

and `devices` API, described in Appendix B. These APIs are filtering the requests from outside, and they deliver each request to its proper handler in each Django application. Requests like login and logout are redirected to `authentication`'s API with the prefix */auth/* in the URL, and requests of devices, sensors and trips are all redirected to the `devices`' API with the prefix */devices/* in the URL.

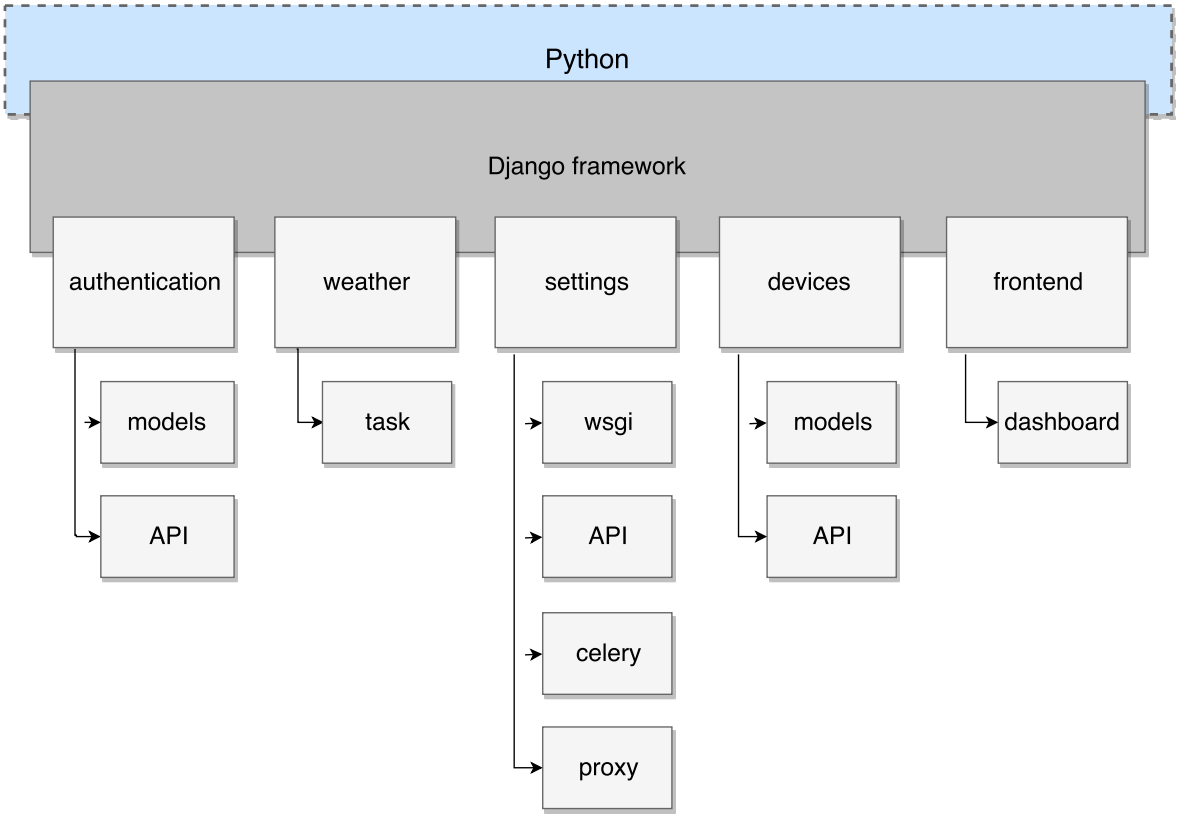


Figure 4.3: Diagram of the backend with Django framework.

The `proxy` application, exists with one simple purpose, to be a proxy for data requests to the IoT Platform. All queries to the platform pass through the backend so that the client never communicate directly with the platform server. Thus, the client cannot exploit the server.

Celery is an asynchronous task queue/job which aims in real-time operation, but also with a very used feature which is the support to tasks scheduling. We integrate this technology in the Django's project, and schedule a task as shown in Code 3, which was created in the task module of the `weather`'s application.

```
# Scheduling task
app.conf.beat_schedule = {
  'run-every-10-minutes': {
    'task': 'weather.tasks.parse_weather_climetua',
    'schedule': 600.0
  }
}
```

Code 3: Celery scheduling code in Django project.

The Celery task in `weather`'s application runs every 10 minutes because the objective of that task is to parse all useful data from the weather website of the CESAM. So, the task gets the web page from `http://climetua.fis.ua.pt/legacy/main/current_monitor/cesamet.htm` and extracts all relevant data one by one to be published through MQTT to the IoT Platform with a JavaScript Object Notation (JSON) message as shown in Code 4. As the website updates its information each 10 minutes, periodic task scheduling should never be less than the same value, to avoid produce duplicate data. The reasoning to this is that there is no API available to obtain weather data from Climetua.

```
   {"timestamp": 1506646896401, "lat": 40.6308333, "lng": -7.3425000,
↪   "temperature": 14.5, "dewpoint": 13.7, "humidity": 95.0, "wind_direction":
↪   "NE", "wind_speed": 0.0, "wind_chill": 14.5, "bar_value": 1022.2,
↪   "bar_prevision": "Steady", "rain_rate": 0.0, "storm_total": 0.0, "thw_index":
↪   14.7, "heat_index": 14.7, "uv_index": 0.0, "solar_rad": 0, "rain_today": 0.0,
↪   "rain_monthly": 8.9, "rain_yearly": 507.0, "temp_high": 14.9, "temp_high_time":
↪   1506639840000, "temp_low": 14.5, "temp_low_time": 1506645660000,
↪   "dewpoint_high": 14.4, "dewpoint_high_time": 1506639840000, "dewpoint_low":
↪   13.9, "dewpoint_low_time": 1506639600000, "humidity_high": 95.0,
↪   "humidity_high_time": 1506639600000, "humidity_low": 95.0, "humidity_low_time":
↪   1506639600000, "bar_high": 1022.9, "bar_high_time": 1506640320000, "bar_low":
↪   1022.2, "bar_low_time": 1506646080000, "rain_high_rate": 0.0,
↪   "rain_high_rate_time": 0, "heat_index_high": 15.0, "heat_index_high_time":
↪   1506639600000, "uv_high": 0.0, "uv_high_time": 0, "solar_rad_high": 0.0,
↪   "solar_rad_high_time": 0}
```

Code 4: JSON format for MQTT messages.

The `frontend`'s application handles all the dashboard functionality. In this application, the Django serves all static files belonging to the website. That means that all requests to the root path of the server give the index web page to the client. After that, the client side becomes in a stand-alone client application and it makes its requests to the server. The dashboard is detailed further in Section 4.4.

4.2.1 DATA MODELS

Data models in Django are one of the things that made us choose to adopt Django for the backend. Django behaves as a middleware between the data model and the storing process. It does not need to work directly with the database, but we have to know how they work and how to make the relationships, primary keys and everything associated with Structured Query Language (SQL). With this feature, we can migrate from one database system to another without changing our data model.

In both `authentication`'s application and `devices`' application, *models* is the module where the data models are defined. These data models represent how the data will be organized in the database whatever it may be.

The `authentication`'s data model is an extension to the default User's model from the Django. It adds more personal information about the user and sets a role to each one. The roles can be admin, staff, user, or deactivated. The purpose of these roles are to give different permissions in the dashboard and this point will be discussed in Section 4.4.

The classes diagram formed from all created data models and contained in Django project is presented in Figure 4.4. The User's model is a built-in data model and comes with the Django's framework. The `device`'s data model is a set of three different models. Those data models are:

Device This data model is responsible for creating the logic for the devices. Each device has a unique Universally Unique Identifier (UUID), and it belongs to one user, thus creating a relationship with the Users' data model. Each user can have as many devices as he wants, but one device belongs only to one user. It is possible to add other information like the type of device (a smartphone or an in-car unit), a name and optionally a description, etc.

Sensor Like the Device's data model, for this data model it was created to store all types of sensors that can be used to obtain data. Whatever it is the device, there are a set of predefined sensors. In this data model are defined several types of sensors, and the user needs to associate one or more sensors to each device. Thus, this data model maps the sensors to each device. The objective is to support devices with different sets of sensors.

Trip The Trip data model gives a quality of experience to the user when we talk about the analyzing of the collected data. To evaluate collected data, is necessary something which gives us the concept of different trips. Instead of having all data together, we created a system of trips. This system is detailed in Section 4.2.2.

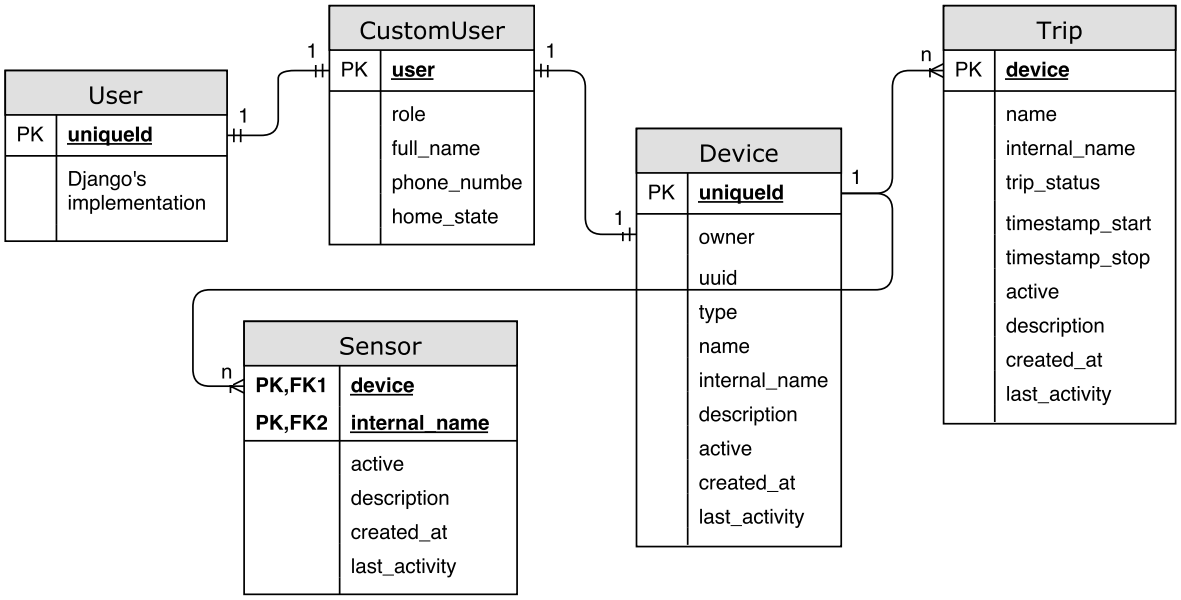


Figure 4.4: Class diagram of the Django project's data models.

4.2.2 TRIPS AND TRACKS

The system of Trips consists in storing the start and finish time of each device's trip. Since we have those times, and a name to identify each trip, the analysis of specific data collected by that device, became much easier.

Every time a device starts running, being an FCD or an In-car Unit, it should initiate by registering a new trip. That register is specified by a JSON message.

Code 5 specifies the message which must be present in the API POST request for Start/Stop a trip. This process requires many validations:

a) It is never possible to stop a nonexistent trip. Firstly, the trip must be created, and only then it should be stopped.

b) The UUID provided in the URL request and the device *id* provided in the POST request, both must be valid and existing in the database.

c) When stopping a trip, the provided timestamp in the POST request can never be before or equal to the starting timestamp.

d) The trip name in the POST request when starting a trip can be anything. But when stopping that trip, the name provided, device id and device UUID must match into only one trip.

```
{
    "id": 1,
    "timestamp": 1499697310000,
    "trip_status": "<Started/Stopped>",
    "name": "UA to Jumbo 2017-07-10"
}
```

Code 5: JSON specification to Start/Stop a trip.

Achieved the Start/Stop trips system, the next step was to implement a method to get the trips information. The methods below of POST and GET have the same URL, thus both provide the UUID of one device.

GET `http://cruise.aws.atnog.av.it.pt/api/devices/trips/<device_key>/`

POST `http://cruise.aws.atnog.av.it.pt/api/devices/trips/<device_key>/`

```
{
    "id": 1,
    "timestamp": 1499697310000,
    "trip_status": "<Started/Stopped>",
    "name": "UA to Jumbo 2017-07-10"
}
```

Every device should create its trips. After doing it, the system will recognize the trip and will show it in the dashboard. It is not necessary that the trip ends to be seen on the dashboard. The system can display the data in real-time, even before it ends.

4.3 TRACKERS

Collecting data from multiple sources is one of the objectives of this dissertation. This section will describe our implementation in this topic. Data collectors can be bought, but they are expensive and with a little of knowledge and work it is possible to build devices that can do almost the same work and at a lower cost. This Section explains the building process of the proposed devices, as well as the systems that actuate on them and its features.

4.3.1 FLOATING CAR DATA

Floating Car Data was the start point to explore the trackers' section. Making a data collector from a smartphone showed up to be a task not so simple to complete. Therefore, this subsection will explain our approach in FCD using a smartphone.

To develop the application was used the Android Operating System because we have more experience in this area but also because the target audience is more accustomed to using Android OS. Accordingly to Figure 4.5, the decision was to develop the mobile application to achieve the maximum people possible. Thus we can see that the biggest slice of the market is the Android version 4.0 and above. For the application, we use the API version 14 which means Android version 4.0 with codename Ice Cream Sandwich.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.4%
4.2.x		17	3.5%
4.3		18	1.0%
4.4	KitKat	19	15.1%
5.0	Lollipop	21	7.1%
5.1		22	21.7%
6.0	Marshmallow	23	32.2%
7.0	Nougat	24	14.2%
7.1		25	1.6%

Figure 4.5: Android OS distribution in 2017-09-11. Versions under 0.1% are not displayed. [30]

In Android OS, when developing to old versions, the lack of usable libraries is significant and becomes a problem. This fact comes because almost all libraries are being developed to newer versions, with more potential, indeed, but many times more computationally demanding. Therefore, some functions were developed from scratch, which is not always the best way, since we can create better applications by using what contributions of the community reducing the time required for implementation so that we can focus on the real problem.

The name given to the application was Metriget and comes from *metrics* and *gather*, and the purpose is to be a data collector. The first thing is what should we collect from a smartphone, among so many different models. So the focus was on the most common sensors incorporated in the smartphones since the year 2011, release year of the Android Ice Cream Sandwich.

The most interesting sensors to log were identified, and a set of sensors were chosen to be logged, which are:

GPS	An obvious metric to log, once the ambit of the solution is traffic monitoring, is getting the vehicle's positioning once it is a primary need.
Accelerometer	Very common sensor since the first smartphones. Useful to measure the device's instantaneous acceleration.
Gyroscope	Another common sensor present on smartphones. Used to obtain the orientation and angular speed of the smartphone.
Magnetometer	A sensor to measure magnetism. Can measure a magnetic material or a change of a magnetic field. Can be used as a compass to detect changes in the direction of an ambient magnetic field.
Pressure	Not so common between the smartphones, but can provide information about the ambient pressure which let us obtain the object altitude, due to pressure changes relatives to the altitude.

All of this sensors are supported by the API version 14 of the Android OS, despite in many low-end smartphones some of the sensors might not be included.

The most critical sensors is the GPS once he will provide significant data, about the positioning of the device. This positioning will be a key value when analyzing the rest of the collected information.

Other sensors will be used to compute the behavior of the smartphone. This information can be useful when comparing with the positioning of the smartphone. Is expected that the smartphone is positioned in a secure and fixed place in the vehicle, using a smartphone holder for instance. Otherwise, the data collected will be affected by external moves and rotations.

DATA ACQUISITION

The logged data comes from all of the sensors listed in Section 4.3.1. This acquisition is not polled all at once, each sensor has its own reading period which can vary from device to device or even from manufacturer to manufacturer.

The Android API provides four reading delays: **a)** SENSOR_DELAY_NORMAL - 200ms **b)** SENSOR_DELAY_UI - 60ms **c)** SENSOR_DELAY_GAME - 20ms **d)** SENSOR_DELAY_FASTEST - 0ms [32]. Our approach was to have all sensors except the GPS, doing readings with a sample rate of 60 milliseconds (SENSOR_DELAY_UI), which is enough for our purpose. To detect these readings, we have an event listener where the value is stored using a Hash-map as a data set. This process performed by a service called *DataService*. A service is a component that can run in the background, even without the focus on the application [31].

Figure 4.6 depicts the modules' diagram which constitutes the application's core. A set of services and activities grant the excellent operation of the application.

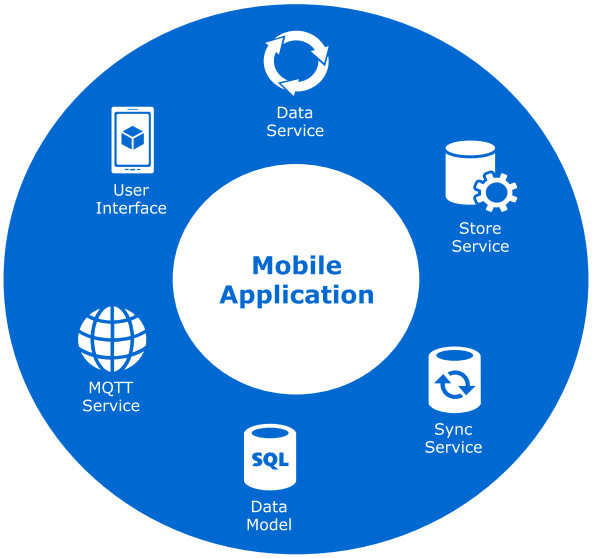


Figure 4.6: Application modules' diagram.

The GPS acquisition works differently. Firstly is used the best location provider available, which should be the GPS provider. If not available, then it is used the Network Provider. This provider determines location based on the availability of a cell tower and Wi-Fi access points, through a network lookup.

The Hash-map contains the most updated data from the sensors but they are in the phone's memory, and it must be somehow persistently saved. Two services were created to store data in two different ways, one called *StoreService*, and the other *MQTTService*.

In the first place, to store locally in a database it is used the SQLite library. This library implements a SQL database engine without depending on a separate database management process [18]. As we did not need a powerful database, the SQLite was the perfect option, despite the fact of being the only option given by the Android API.

The service *StoreService* stores the most updated data available in the Hash-map every 250 milliseconds (a frequency of 4 Hertz). We have tested with a higher frequency, and it makes the application to freeze and work slower due to the consecutive writings in the internal's phone memory, which is by far much slower than the phone's Random Access Memory (RAM) which is used by the Hash-map.

To make the system more useful, we would like to have the information in real-time for instant analysis. Another useful feature is if the smartphone has no internet connection at that moment, it should give the possibility to synchronize later the data collected with the IoT Platform. Using the IoT Platform makes that possible. The service *MQTTService* uses the MQTT protocol, which will publish all data to the platform. In this service is used a publish rate equal to the *StoreService*'s service.

Every time the service publishes data, it sends a JSON message as shown in Code 6. This message defines all sensors of the data set and the timestamp of the capture.

```
{
    "ACC": {
        "X": -1.94, "Y": 5.77, "Z": 8.18
    },
    "GPS": {
        "lat": 40.631123, "lng": -8.659337
    },
    "GYR": {
        "X": -0.13, "Y": 0.12, "Z": -0.07
    },
    "MAG": {
        "X": -4.3, "Y": -31.74, "Z": -14.38
    },
    "MOTION": {
        "pitch": 86.49,
        "roll": 148.52,
        "yaw": -95.82
    },
    "PRE": 1002.549,
    "timestamp": 1506464632384
}
```

Code 6: JSON format for MQTT messages.

The service *DataService* computes other data that cannot be provided directly by the sensors. This data represents the behavior of the smartphone, which uses the built-in functions provided by the Android API, to get the orientation of the smartphone.

MOBILE APPLICATION

The aim was not to have a fancy application but to have a functional one. When opening the application is presented a screen as depicted in Figures 4.7, 4.8, and 4.9. The empty red circles on the top, identified by the number 1, will give the feedback to the user about the existent sensors that can be used to log data. They will become green and with a check symbol to quickly see which are being logged. The smaller circles at the bottom, identified by the number 2, act as status lights. From left to right: `Ready` means the device is logged in system; `Device` shows if the device was identified by the system; `Trip Status` shows if the trip status is correct; `Publish Status` indicates if the device status was publish with success.

The menu displayed on the right corner, identified by the number 3, contains the application settings. These configurations are used to help in the use of the application and are following described:

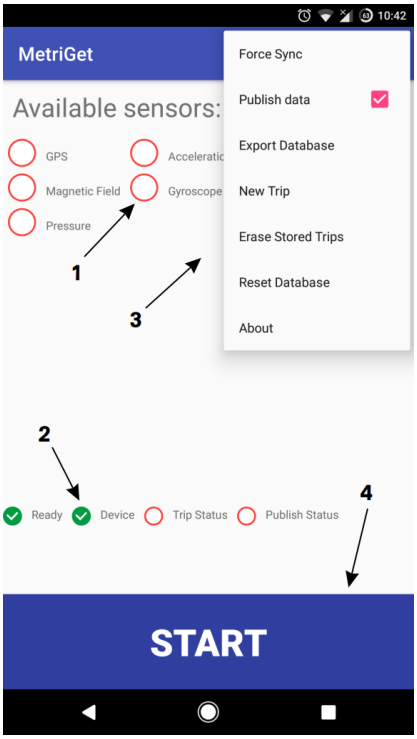


Figure 4.7: Main activity screen.

Figure 4.8: Application in running mode.

Force Sync This option is responsible for synchronizing offline data stored in the database that was not published to the IoT Platform. This feature is useful when there is no internet connection and later is possible to publish the data to the platform when the internet connection is available.

Publish data This option controls if the application should publish or not the logged data.

Export Database If we need the logged data still stored in the application, this option will export all data from the database to a file in Comma-separated Values (CSV) format.

New Trip To start a new trip, or give a name to it. The application gives a default name for the trip, but we can choose one name to be more descriptive to help when searching for the smartphones' trips in the dashboard.

Erase Stored Trips Internet connection may not always available. Therefore the application will store the trip status locally to be published along with the logged data when the user triggers the force sync feature. This option will erase only the stored trips from the database.

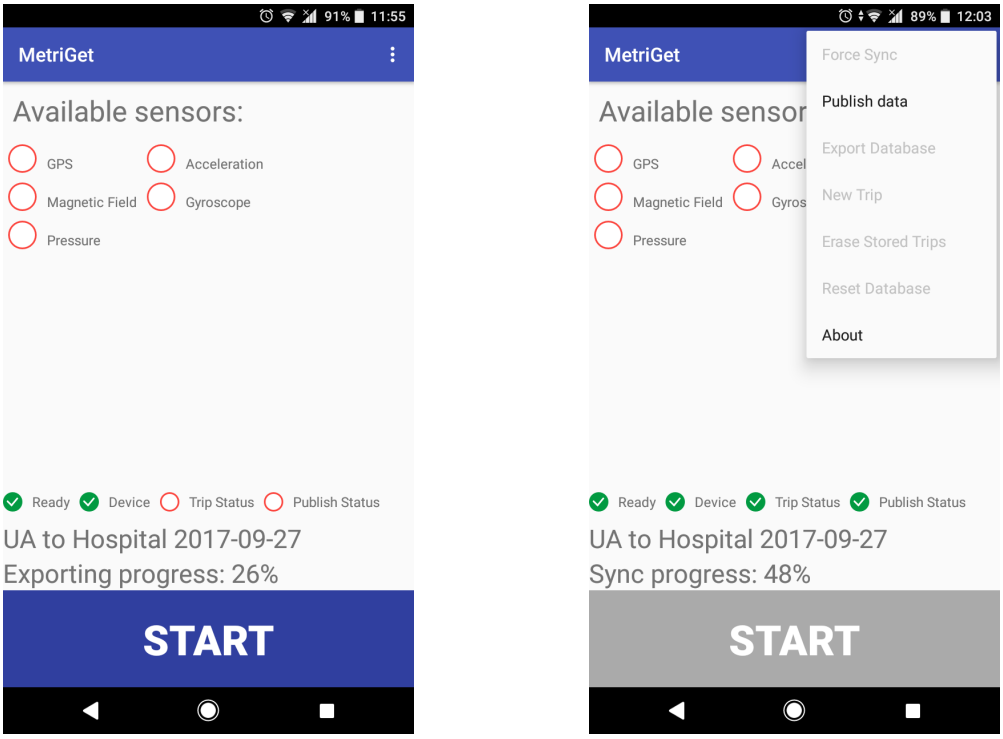
Reset Database We can always clean our database using this option. All the

collected data stored in the database will be erased. Here does not include the trips.

About This last option takes the user to another activity where is shown a brief description of the application, the version, the authors, and a contact.

Start button on the bottom of the application, identified by the number 4, starts the logging service. When in running mode, the application looks like the Figure 4.8. Right below the indicator lights, identified by the number 5, we will see the trip name and the starting date and time. An advantage of this application is the capability of working in the background, significantly reducing the power consumption. This property comes with the service component.

When a car goes through a tunnel, is almost impossible to have an internet connection, unless it is provided by underground antennas, as well as GPS signal. For the GPS signal, there is no any solution previewed. Although, for the problem of network failure, the data can be exported or synchronized later as shown in Figure 4.9. Figures 4.9a and 4.9b displays the progress of the action that they are executing. Some of the controls are disabled while these tasks are running.



(a) Exporting screen. **(b)** Synchronizing screen.

Figure 4.9: Feedback to user about the progress when exporting and synchronizing data.

4.3.2 IN-CAR UNIT

This section will detail the prototype developed to log data, like the FCD but more precise and complete. By now the mobile application is ready to collect data and publish them. Now, we aim for a bit more capable device, that can log data, which is more precise and that could get data through an OBDII port, from any vehicle exposing a Crontoller Area Network (CAN) data bus.

Joining the behavior values with the GPS positioning and metrics from the vehicle's data bus will be a big step to achieve our objectives. This device uses a Raspberry Pi 3, that is like a conventional computer but in a smaller size (like a credit card), and of course with much less computation power and memory. This kind of devices has a relatively low cost compared to what can offer to us. With a quad-core CPU at 1.2GHz, 1Gb of memory, Bluetooth 4.1, 802.11n wireless, and a price of 35€ are some of the characteristics that make it the perfect device to implement our solution of the In-car Unit.

The Raspberry Pi does not have any motion sensor embedded, and we need sensors like the accelerometer or gyroscope. This Raspberry Pi comes with 40 General Purpose Input/Output (GPIO) pins, and those pins are used to connect and control external hardware, like a Light Emitting Diode (LED) or a button. With this in mind, it is possible to integrate the motion sensor into our Raspberry Pi using these GPIO pins.

Starting by weld the sensors to one prototyping board. This board will be docked in the GPIO pins of the Raspberry Pi.

Figure 4.10 is the final result of the prototyping board with the welded sensors and a GPS module. There are several components in the figure and each one with different purposes, where they all have its name printed, except one, so we will detail each one addressing those names.

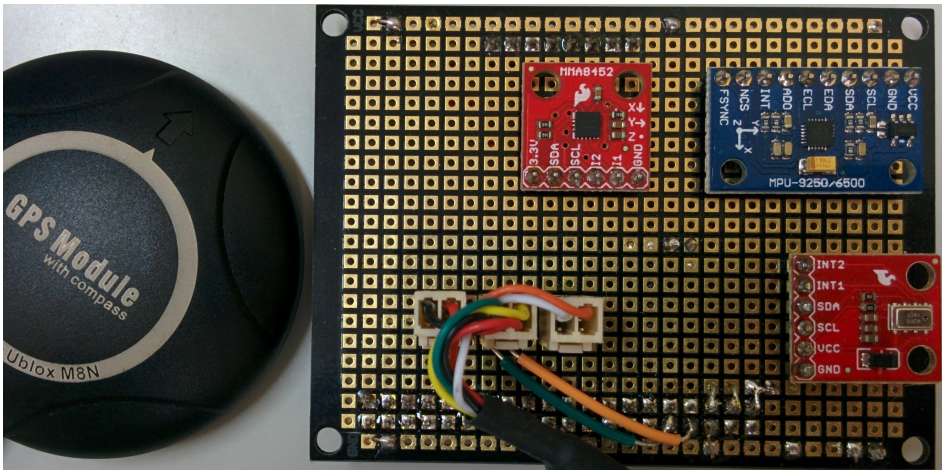


Figure 4.10: Prototyping board with welded sensors and the GPS module.

Ublox M8N This component is a GPS receiver. It has concurrent reception of up to 3 Global Navigation Satellite System (GNSS) which is a system of satellites (GPS, Galileo, GLONASS, BeiDou), that provide geospatial positioning around all the globe. This module has also integrated a compass. This component cost between 20-40€ including the compass.

HMC5983 A 3-Axis Magnetometer for magnetic sensing. With many configurations available and supports communication with Inter-Integrated Circuit (I^2C) or Serial Peripheral Interface (SPI) serial bus, and with 220Hz of the maximum output rate.

MMA8452Q Is a single electrical component easily usable, which has a smart low-power, 3-Axis Accelerometer with 12 bits of resolution. Selectable full scales of $\pm 2g / \pm 4g / \pm 8g$ with an high-pass filter, and a I^2C interface for communication. The cost of this component is about 9€.

MPU-9250_6500 A 3-Axis Gyroscope (plus 3-Axis Accelerometer and 3-Axis Magnetometer) motion tracking device. With a programmable gyroscope full-scale range of $\pm 250 \pm 500 \pm 1000$ and $\pm 2000°$/sec (dps), a programmable accelerometer full-scale range of $\pm 2g / \pm 4g / \pm 8g / \pm 16g$, and a full-scale magnetometer range of $\pm 4800 \mu$T. Supports I^2C interface for communication. The cost of this component is about 8€.

MPL3115A2 It is a board pressure sensor that provides altitude data up to 30 cm of precision. The data is digitalized to a resolution of 24 bit. The pressure output is given in Pascal unit, and the altitude can be resolved in fractions of a meter. This device also provides the temperature in degrees Celsius with 12 bits of resolution. Communication interface I^2C. The cost of this component is about 12€.

With just about 100€ plus the OBDII adapter, it is possible to have a complete device set capable of collect precise data.

Now having all the components on the prototyping board we just dock it on the Raspberry Pi. After we activate the I^2C interface on the Raspberry Pi, the interface will be then listed in */dev/* directory. Installing the I^2C tools from the package *i2c-tools* we can list all I^2C interfaces connected to our Raspberry Pi.

Once we have a ready to work prototype, is necessary a program to collect all data from these different components. To develop this program was used Python language version 2.7.12. In first place the program will try to connect to the OBDII port, so it searches for valid USB and RF ports.

The OBDII gives us access to the information that flows in the CAN data bus of any vehicle exposing it. We do this using an OBDII interface through a Bluetooth connection as in Figure 4.11. This adapter is paired with the Raspberry Pi, then in the program, we can access it through an open source library.

Figure 4.11: OBDII adapter with Bluetooth connection.

The purpose of adding a data source like this is that this is the perfect information that we could know to our scenario. Not only the behavior, positioning, or altitude, but the state how the vehicle behaves. As we also want to identify bad points in the city, which generate traffic jams, we also need to know what are the consequence inflicted to the driver when he is in those traffic queues for instance. This data will provide that information, when interpreting the vehicle's behavior, so that it can cross that information with the positioning in the city, the weather state at that time, and vehicle's behavior, then we will try to identify driver's behavior and characterize him somehow.

To help us with this task, we choose some of the most shared and supported commands. The problem with OBDII interfaces is the great variety of supported commands changing from car to car. Even cars from the same brand from one model to another have different available commands. To work around this problem, the solution was to read only the most supported commands by vehicles. Those commands are listed below:

Fuel Level Get the fuel level in the tank measured by the sensor. This command gets this metric in percentage.

Speed The current speed of the car, just like displayed in the speedometer. This command gets this metric in kms/h.

RPM	Number of times the crankshaft rotates about itself, so it is the frequency of the rotation per minute. Useful metric to include in the driver pattern determination. This command gets this metric in Revolutions per Minute.
Engine Load	Every external force applied to the engine. If connected to nothing, the load is nearly none regardless of throttle opening or RPMs. A key metric used to characterize the driver behavior. This command gets this metric in percentage.
Coolant Temperature	Measures the temperature inside the engine. Useful to understand if the engine is overheating, or if the driver starts to pull too much from the engine when cold, thus it might be possible to detect situations of stress due to rush hours or a rushed driver. This command gets this metric in degrees Celsius.
Run Time	Elapsed time since the engine starts running. This command gets this metric in seconds.

Interesting commands like throttle position, or Mass Air Flow Rate (MAF) would be really helpful, but this is out of our reach. This way we might guarantee the compatibility with more vehicles.

A configuration setup is needed before starting reading values from the sensors. The compass sensor inside the Ublox M8N was selected to continuous measurement mode. The MMA8452Q accelerometer is configured with an 800Hz output data rate and with a full-scale 8g. The MPU-9250_6500 in the gyroscope sensor was setup with a full-scale 2000dps and the accelerometer was setup with a full-scale 16g. The MPL3115A2 pressure sensor was setup with an oversampling ratio of 128 and in altimeter mode. Finally the GPS module will be running in a separate thread inside a continuous loop until stopped.

The connectivity of the program is not a requirement, once he has a mechanism of safe publishing. Once more it is used the IoT Platform to publish the data. With a safe publish method, it can store locally the collected data using a SQLite database and the publish mechanism will send the data as soon as possible, once it has internet connectivity. The mechanism is implemented with a priority queue, so it will first publish the most recent collected data.

The collected data is then aggregated all in one JSON message as in Code 7 and published through MQTT to the platform.

```
{
    "GPS": {
        "alt": 115.5,
        "lat": 40.6981588,
        "lng": -8.5091518,
        "time": "2017-06-01T17:20:36.000Z"
    },
    "FUEL_LEVEL": 38.43,
    "SPEED": 73.0,
    "RPM": 1676.25,
    "ENGINE_LOAD": 21.18,
    "COOLANT_TEMP": 89.0,
    "RUN_TIME": 1239,
    "VSP": 182.623,
    "VSP_MODE": 14,
    "CFangleX": 37.94,
    "CFangleY": -11.52,
    "HEADING": 21.53,
    "TILT": 21.53,
    "PRESSURE": 100.141,
    "TEMPERATURE": 26.12,
    "ACCELEROMETER_MMA": { "X": 0.0, "Y": 0.0, "Z": 0.0 },
    "GYROSCOPE": { "X": 0.0, "Y": -101.75, "Z": 0.0 },
    "MAGNETOMETER": { "X": 139.4, "Y": 55.0, "Z": -111.8 },
    "timestamp": 1500469079000
}
```

Code 7: JSON format for MQTT messages.

As it can be seen in Code 7, after the GPS and engine metric appear the *CFangleX*, *CFangleY*, *HEADING*, and *TILT*. These four stats bring the vehicle's behavior calculated by the program at runtime. This data is computed using the raw data from the sensors and after it is applied the Complementary Filter (CF), as explained in subsection 4.3.2.

It is included a calculated metric at runtime called VSP. The Vehicle-Specific Power subsection will detail this information.

To have redundant data storing it was implemented a data logger in the program to store the collected data, in CSV format. This logger will always be recording, and it does not work as the safe publish, which only saves when publishing is not possible.

LIFE CYCLE

Before start reading and logging data, a set of provisioning needs to be done. Figure 4.12 shows the life cycle of the program. When the Raspberry Pi initializes, it automatically starts the program. The program follows a sequence of well-defined tasks until it reaches the main loop. These tasks are: **a)** The first thing to do is trying to connect to an available OBDII port; **b)** After task *a)*, if it will not succeed, it just

continues, but it will not be able to collect vehicle's data. However, the next step is to set up the sensors, where they will be configured to behave as we want, with the rates and scales which fit better to the solution. **c)** Initiate the MQTT Client and the safe publishing mechanism, and then connects to it. **d)** Initiate the GPS module to be configured in stream mode, which will run on a separate thread to continuously update the vehicle's positioning. **e)** Starts a new trip, using the backend's API. The name of the trip will be automatic and is composed by the date and time of creation. **f)** Lastly, we start the OBDII readings. This will be in watch mode, which means that the readings will always be auto-updated.

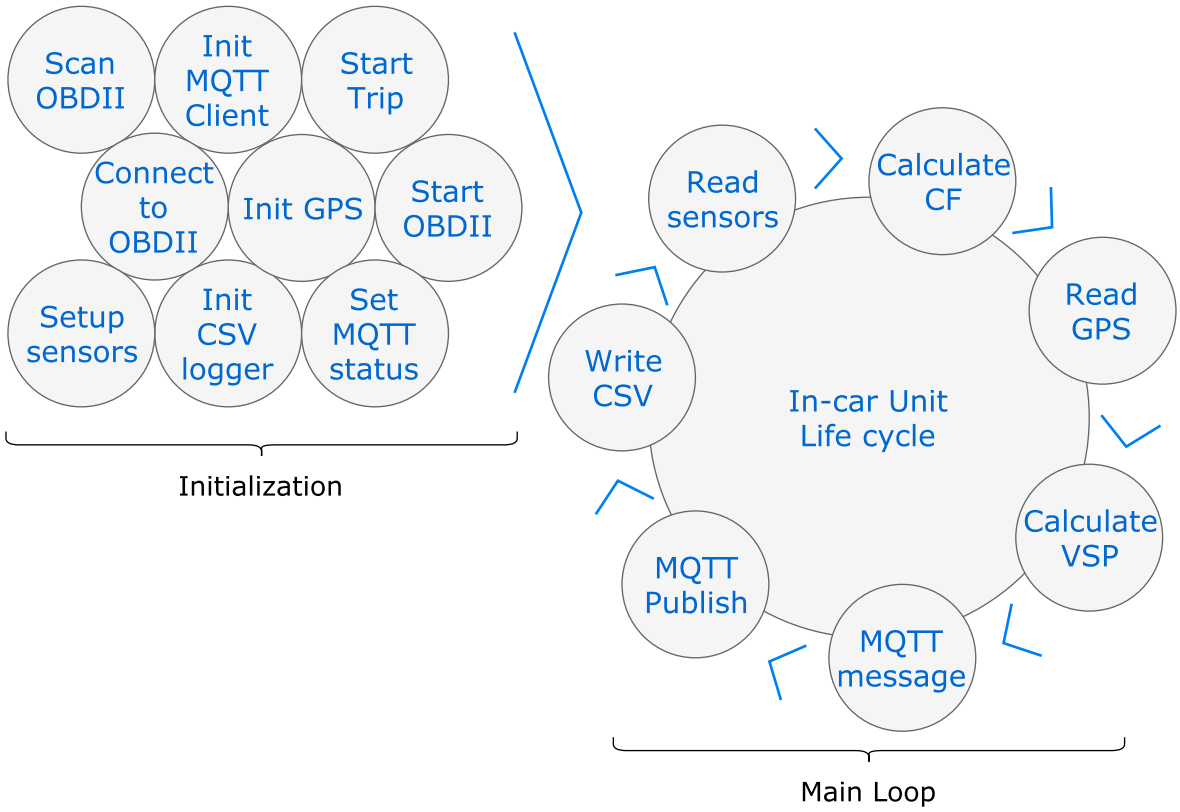


Figure 4.12: In-car Unit's life cycle.

After prepared and initialized all systems, the main loop can start. In this loop, is possible to read the raw data from all motion sensors, the gyroscope, accelerometer, and magnetometer. With this reading, we will compute the behavior of the vehicle by applying the CF. When calculated the orientation, the program follows to the GPS reading, where it will read the latitude, longitude, altitude and time. Now that we have a precise altitude, given from the GPS, we can compute the VSP, once it needs the slope deviation from one location to another, which is obtained from the altitude provided by the GPS.

When the readings and calculations finish we must publish and store the information. As an important task, it should be thought on how can we interpret this data. So with this in mind the program builds a JSON message as Code 7 to assure a clear and perceptible message will be stored by the IoT Platform, and then obtained from the dashboard. The same method was adopted when exporting to the CSV file. This format is well-defined and we just need to follow its rules. The first line of the file defines the columns like a table, then we write line by line all data by order and separated by commas.

A set of actions must be performed when stopping the program. These actions are to close the CSV file, stop and close the OBDII connection, stop the trip in the backend, disconnect from the MQTT, stop GPS readings, and lastly put the sensors in standby mode.

VEHICLE-SPECIFIC POWER

Logging pollutant emissions from the vehicles it can help us to analyze the vehicle's behavior, as well how the driver behavior, because the VSP and emissions are correlated. We can assume that the driver's behavior has direct influence in vehicle's emissions [37]. VSP is derived from speed measurements, acceleration, and grade. Therefore, is widely used in the emissions modeling, once this metric can have direct relationship to fuel consumption [28]. Usually, the speed parameter is obtained every second, which is possible for us using the OBDII port.

$$grade = \frac{alt\_slope}{v} \times 100(\%) \tag{4.1}$$

To obtain the grade variable, it is necessary to calculate the vertical rise/horizontal distance as shown in equation 4.1. The alt_slope is the altitude variation relative to the last altitude value, and the v variable is the instantaneous velocity in meters per second. This parameter, represented in percentage, is the altitude variation from one second to the following.

$$VSP = v \times \left[1.1 \times a + 9.81 \times \sin\left(\arctan\left(grade\right)\right) + 0.132\right] + 0.000302 \times v^3 (kW/ton) \tag{4.2}$$

To calculate the VSP we used the formula given by the equation 4.2 that was firstly introduced by [37] in 1999. We expect to have a relationship between VSP and engine load or vehicle's speed. The formula gives us a value expressed in $kW/MetricTon$,

where v is the velocity in meters per second, and a is the acceleration in meters per second squared.

The VSP is then converted into a 14 mode range [9]. According to the Table 4.1, we can set a VSP mode, and thus provide an easier way to analyze this parameter in the dashboard.

Table 4.1: VSP mode relation.

VSP	VSP Mode
]-∞; -2[	1
[-2; 0[	2
[0; 1[	3
[1; 4[	4
[4; 7[	5
[7; 10[	6
[10; 13[	7
[13; 16[	8
[16; 19[	9
[19; 23[	10
[23; 28[	11
[28; 33[	12
[33; 39[	13
[39; +∞[	14

COMPLEMENTARY FILTER

An Inertial Measurement Unit (IMU) is a set of sensors responsible for keeping track of the device's orientation, for instance, to automatic control of the screen tilting [65]. In our case, the IMU that we are working with are the sensors installed in the In-car Unit.

When thinking in doing an IMU-sensor a commonly used technique is the Kalman Filter. This filter relies on "recursive properties, and its status as the optimal estimator for one-dimensional linear systems with Gaussian error statistics"[26]. It is an algorithm that allows an exact conclusion in a linear dynamical system [26]. However, this option has some limitations, like the inherent complexity and the required computational capability. Once we aim at devices with limited computational resources, it is necessary to use an alternative solution: the Complementary Filter.

CF have been studied for a long time in IMUs applications since its designing by Wirkler in 1951 [46]. The approach used to obtain the object's behavior is based on fusing two different sources as explained further in this Section. Furthermore, many

variants of the CF algorithm are discussed among authors where they compare it with the Kalman Filter and combine different techniques to compute behavior.

The IMUs that are used for both devices have 6 Degrees Of Freedom (DOF). This means that we have a 3-Axis Accelerometer, and a 3-Axis Gyroscope. A 6 DOF IMU was supposed to be capable of measuring the position and orientation of the object it is attached to.

To compute the device's behavior using the CF is necessary both accelerometer and gyroscope data. It is needed to compute the angular position of the device. By integrating the angular velocity over time from the gyroscope data and from the accelerometer data is possible to determine the position of the gravity force [65]. Regarding this, we still have difficulties in obtaining the correct angular position of the device, as described below.

Accelerometers This kind of sensors measures the forces acting upon the object. They sense the gravity force and dynamic forces like sudden start/stops. Consequently, it measures forces that we want and others that we do not want to. This external forces will generate an amount of noisy readings which will disturb the measurement. This sensor is only reliable when used in the long-term, which means that we need to apply a low-pass filter, to clear distinctive values.

Gyroscopes Obtains the angular position from a gyroscope by measuring the rate of rotation, although it still has the same external forces inducing an error, like with the accelerometer. As we integrate over time, we start to get a drifted values, so even if the object goes to the original position, our measured value does not go to the initial value. A gyroscope gives us reliable measurements in short-term, otherwise, we will get drifted readings.

Before using the CF, the retrieved data from the accelerometer and the gyroscope needs to be converted to proper angles, once they are in raw values.

To compute the accelerometer angles we use trigonometry as shown in Code 8. Using the arctangent function with the raw values from the opposite accelerometer axis, we get the tangent value of the other angles. This function gives us the angle expressed in radians, so we need to convert it to degrees by multiplying it by the constant $\frac{180}{\pi}$ represented by the variable *RAD_TO_DEG*. The variable *M_PI* is the constant value of π.

```
# Convert Accelerometer values to degrees
AccXangle = (math.atan2(ACCy, ACCz) + M_PI) * RAD_TO_DEG
AccYangle = (math.atan2(ACCz, ACCx) + M_PI) * RAD_TO_DEG
```

Code 8: Convert Accelerometer raw values into angles.

For the gyroscope, for each axis, is converted the raw value to degrees per second just by multiplying the raw value with the appropriate gain. This gain is defined by the manufacturer and is described by the device's datasheet according to the sensitivity level set on the sensor. After we get the degrees per second next step is to calculate the angles. As explained, the gyroscope should be tracked over time. So we periodically calculate the angle by adding the last known angle with the new value of degrees per second times the period size as shows the Code 9. The *LP* variable is the loop period, and it represents the time to complete a cycle of the main loop where the angles are being calculated.

```
# Convert Gyroscope raw to degrees per second
rate_gyr_x = GYRx * G_GAIN
rate_gyr_y = GYRy * G_GAIN
rate_gyr_z = GYRz * G_GAIN

# Calculate the angles from the gyro
gyroXangle += rate_gyr_x * LP
gyroYangle += rate_gyr_y * LP
gyroZangle += rate_gyr_z * LP
```

Code 9: Convert Gyroscope raw values into angles.

The CF helps to solve these problems, by taking advantage from the best of each approach. As we see with the accelerometers, is possible to have reliable measures in long-term only, but for short-term measures, we will use the readings from the gyroscope.

$$angle = 0.98 \times (angle + gyrData \times dt) + 0.02 \times (accData) \tag{4.3}$$

The equation 4.3 shows how the CF is applied to join both accelerometer and gyroscope angles. It takes 98% of the current value of the accelerometer data and adds 2% of the new angle calculated by the accelerometer.

```
# Complementary filter constant
AA = 0.98

# Complementary filter used to combine the accelerometer and gyro values
CFangleX = AA * (CFangleX + rate_gyr_x * LP) + (1 - AA) * AccXangle
CFangleY = AA * (CFangleY + rate_gyr_y * LP) + (1 - AA) * AccYangle
```

Code 10: Applying the CF.

Code 10 depicts the resultant variables to be used. This calculation using the CF has to be implemented inside a loop function, running in the main loop of the In-car Unit. These variables can be called by *pitch*, and *roll* respectively *CFangleX*, and *CFangleY*.

CFangleX, and *CFangleY*, are the results from the fusing technique process for the X and Y axis, that can be named as *pitch* and *roll*. Combining both measurements the drift noise is almost all filtered out from the result.

From these two variables, it is possible to have information about the inclination angle of X-axis and Y-axis. We went further and tried to get information about the *Tilt* angle and the *Heading* direction.

```
# Calculating Heading
heading = 180 * math.atan2(MAGy, MAGx) / M_PI

# Convert heading to be between 0 and 360
if heading < 0:
    heading += 360
```

Code 11: Calculating Heading direction.

To calculate the *Heading* it was used the method in Code 11, where it gives us the direction that vehicle is pointing, through the *atan2* function, and then convert the result to degrees by multiplying by 180 and then dividing the result by π [55].

The *Tilt* is calculated using the accelerometer and magnetometer readings. The method normalizes the accelerometer values, then calculates the *pitch* and *roll*. Once we get these two variables, we can calculate the *Tilt Compensated* value as in Code 12.

```
    # Normalize accelerometer raw values
    accXnorm = ACCx / math.sqrt(ACCx * ACCx + ACCy * ACCy + ACCz * ACCz)
    accYnorm = ACCy / math.sqrt(ACCx * ACCx + ACCy * ACCy + ACCz * ACCz)

    # Calculate pitch and roll
    pitch = math.asin(accXnorm)
    roll = -math.asin(accYnorm / math.cos(pitch))

    # Calculate the new tilt compensated values
    magXcomp = MAGx * math.cos(pitch) + MAGz * math.sin(pitch)
    magYcomp = MAGx * math.sin(roll) * math.sin(pitch) + MAGy * math.cos(roll) -
↪   MAGz * math.sin(roll) * math.cos(pitch)

    # Calculate tilt compensated heading
    tiltCompensatedHeading = 180 * math.atan2(magYcomp, magXcomp) / M_PI

    # Convert tilt to be between 0 and 360
    if tiltCompensatedHeading < 0:
        tiltCompensatedHeading += 360
```

Code 12: Calculating Tilt-Compensated Heading.

This results in a *Tilt-Compensated Heading* value, and the term compensated lead us to a solution to correct a deviation/drift of the *heading* when the magnetometer is tilted [55]. Without the *heading compensation*, the *Tilt* would deviate. Thus, we apply the same formula as in Code 11, to the *Tilt* but now with the compensated values.

The final result of this mathematical formulas contributes to the estimation of the car's behavior. Since the driving pattern can be determined with the help of the accelerations and braking generated by the driver, it is possible to obtain such information in order to help in the driving characterization.

4.4 DASHBOARD

The data sources existent in the solution, provide data from multiple sensors, and with different purposes. As a multiple data source system, it should have a proper way to analyze the data. For our solution, the backend is responsible for serve the dashboard website. So, we maintain a centralized system and take advantage of the excellent features given by the Django framework. Therefore, we have three primary systems working together, and those systems are the backend which is divided into two main parts, the core system and the dashboard, the IoT Platform, and finally, the trackers that fall into two types, the Smartphone, and the In-car Unit.

The dashboard, as a component of the backend, can make use of the User's session in Django. Thus, we can take advantage of the created users' roles, to implement an access control system for the website.

The core of this web application is AngularJS in version 1.6.4, which is a JavaScript framework. The AngularJS makes the typical Hypertext Markup Language (HTML) pages much more responsive, readable, and it is an advantage for the programmers because it is quick to develop.

The frontend can be accessed through the root URL of the server where it was deployed. When accessing the website, the Django will give the index page of our web application. When on the home page, the user can access to the pages listed in the menu. To access other pages, besides the home page, the user must log into the system. Otherwise, it will not be possible to access those pages. Here, the roles will be used to give different access permissions to each user.

After the user logs into the system, he will see the dashboard home page as shown in Figure 4.13. The home page provides an overview of the devices that are online/offline at that moment, also the date time of the last received message, and some statistical information about the number of devices and the count of each type too.

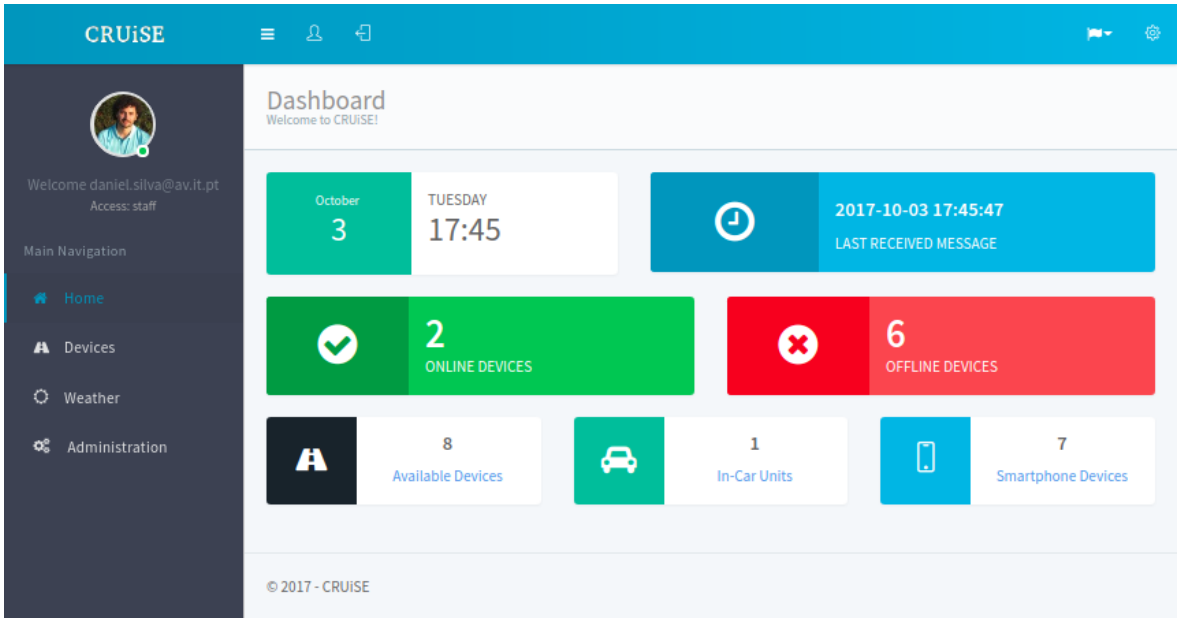


Figure 4.13: Dashboard home page.

Figure 4.14, shows the page where the user can analyze the existing trips of each device. On top of the image, there is a list where it is possible to select the device by type or name. Consequently, a drop-down menu appears showing all trips made by that device. Each trip can be identified by its name, and it also shows a duration time in

minutes. Once the trip is selected, the web application will fetch the data belonging to it and will show them on a map and in charts.

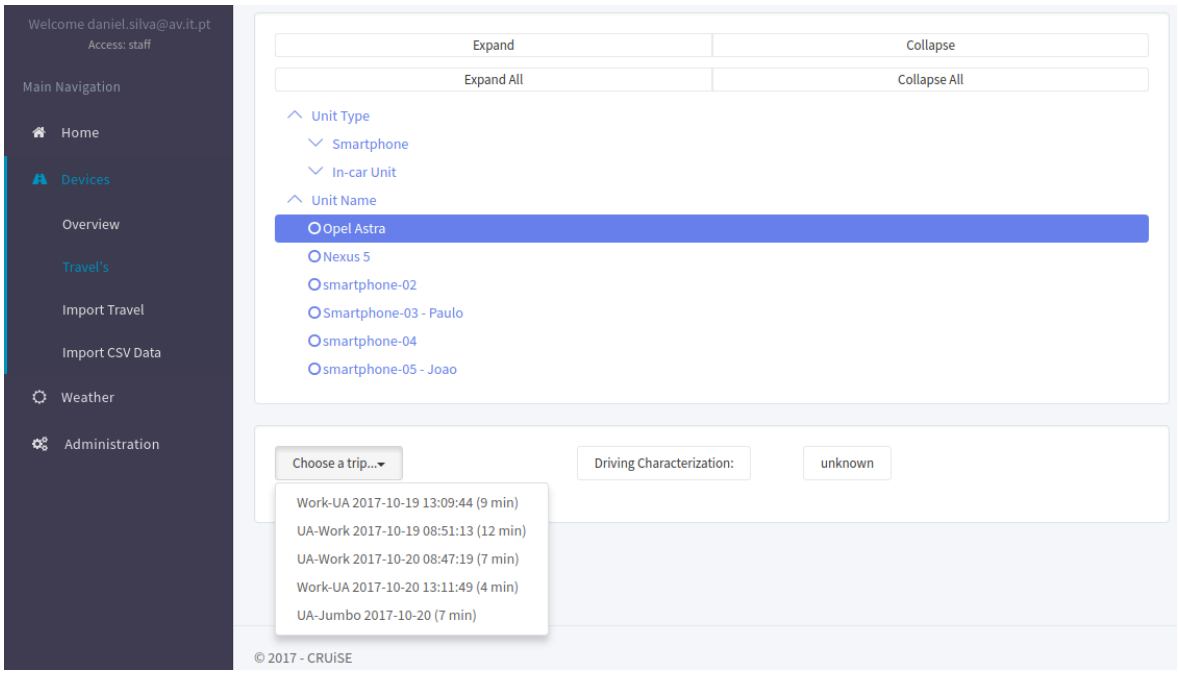


Figure 4.14: Trips listing page.

In the first place, after choosing the device, the web application will obtain all trips made by that device, using the backend API to obtain it. Thus, the user can select the trip he wants to see by clicking on its name. Here, at this point, the web application will automatically fetch the collected data belonging to the selected trip using the backend proxy. The backend receives the request from the client side and redirects it to the IoT Platform where the data is stored. The URL used to make the request will trigger a particular query into the database. The query must identify the device, the timestamp interval, and other parameters needed, as detailed in Section 4.1.

In Section 4.4.1 is detailed the driving characterization algorithm. It is presented an approach on driving pattern classification using a manual algorithm.

When selected the trip, a map is shown, if the GPS coordinates were logged, with a trip path. The map backend adopted to display the GPS tracking was the OSM. This map belongs to a collaborative project to create an open map and free to use without legal concerns and rights.

When passing the mouse over the path line, a pop-up message is shown with information at that point on the map, as depicted in Figure 4.15. That information contains the complete date (date and time), the speed, the revolutions per minute, the altitude, and the VSP mode. Thereby, the user can see some of the metrics directly on the map, and right in the location where the values were collected/calculated.

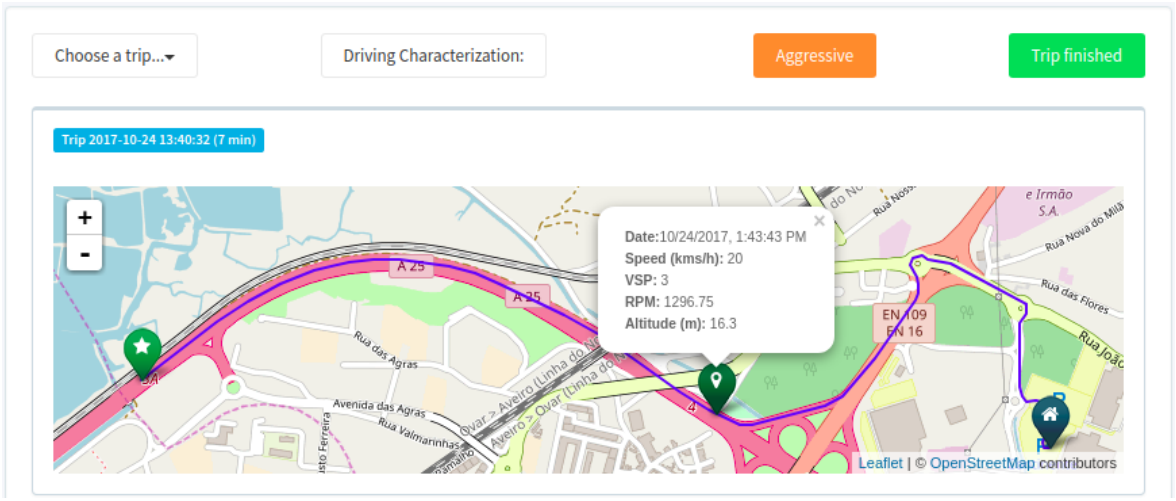


Figure 4.15: GPS path on the OSM map, driving characterization classification, and trip status.

The trip start and stop positions are distinct on the map, and they are distinguished by two markers. To represent the start point, the icon is a house, and to identify the stop point, it is used a star, to be similar to a destination point.

Once the map is loaded, the remaining charts will be displayed as well. The graphs are built using an external library, called *highcharts*, and provides great features and configurable options. They are configured to offer the option of view by period, as depicted in Figure 4.16, so we can choose to display only a time window of ten minutes, for instance; is also configured to be possible to export to some known formats, like Portable Document Format (PDF), Portable Network Graphics (PNG), and others; and is included a navigator bar at the bottom of the chart to help on navigation of the graph when zoomed.

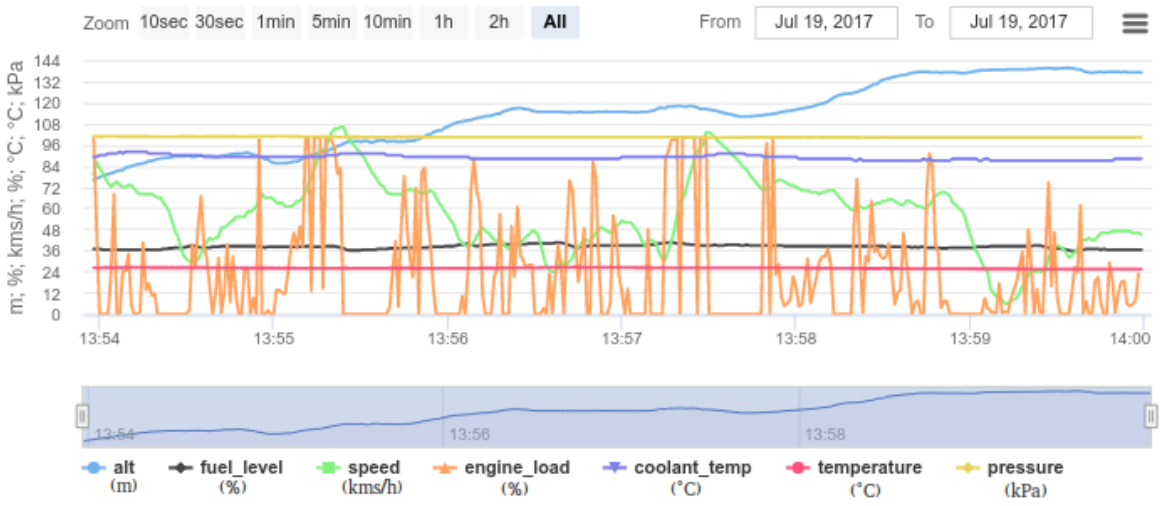


Figure 4.16: Vehicle's data chart.

In first place is shown the vehicle's data chart. In this graph we can find the following parameters:

- Altitude (meters)
- Speed (kms/h)
- Coolant Temperature (degrees)
- Pressure (kilo Pascal (kPa))
- Fuel Level (percentage)
- Engine Load (percentage)
- Temperature (degrees)

In this first chart, depicted in Figure 4.16, we joint the parameters gathered from the OBDII with the altitude, pressure and ambient temperature.

It was not possible to have all the parameters together in one chart. It could be done, but it would be difficult to read and analyze, besides the computational requirement that it would demands from the client machine when processing all the information in one chart only. The contrasts between the parameters values, like temperature and RPM, where the temperature will be almost a constant value close to zero, and the RPMs are always in the thousands, would create such a difficulty in interpreting the graphics. On the other hand, joining these metrics can be very valuable, giving the user the opportunity to compare between different parameters that belong from distinct sensors.

So, further charts were created to avoid charts with too much information and/or with discrepant values.

The chart depicted in Figure 4.17, shows only the RPMs. No other metric fits with the RPM because it is the only one who has values so high.

Figure 4.17: Vehicle's RPM data chart.

The next chart, depicted in Figure 4.18, shows the VSP data. This chart shows both VSP value and mode. The VSP value is the calculated Vehicle-Specific Power that was detailed in Section 4.3.2, and the VSP mode is the converted value on a scale

of 14 modes. Figure 4.18 is an example of how the calculated value is transformed into a range of 14 modes. It becomes easier to interpret the resultant value. Instead of working with the line in blue it is possible to analyze only values from zero to fourteen, represented by the lines in black.

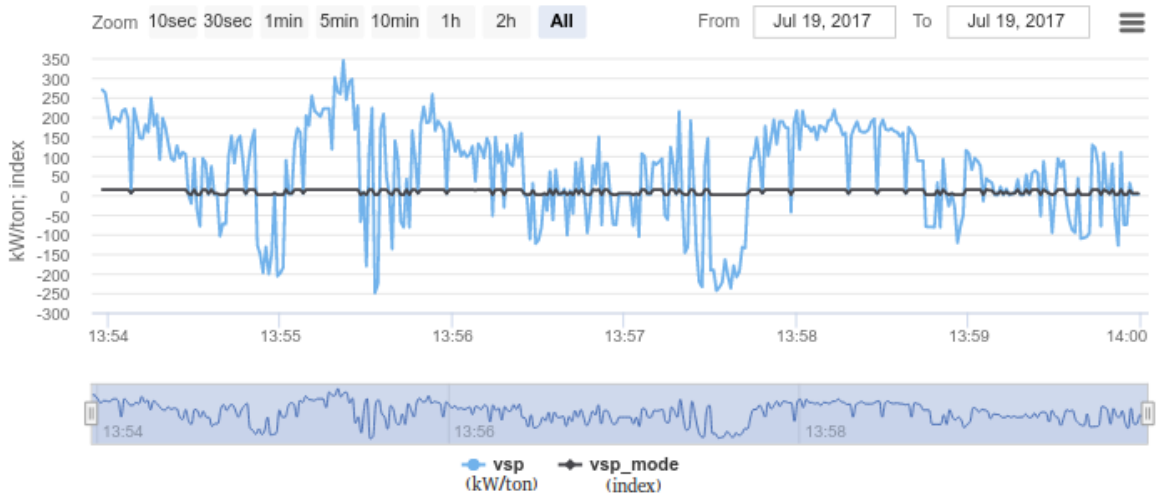


Figure 4.18: Vehicle's VSP data chart.

Turning now to a different type of information, the vehicle's behavior is depicted in Figure 4.19. It is a very challenging chart to analyze, due to the difficult to interpreting the values and of visualizing the orientation/inclination of the car, thus it is a premature approach concerning this subject. This chart is composed by the following parameters:

- *CFangleX* (degrees)
- *CFangleY* (degrees)
- *Tilt* (degrees)
- *Heading* (degrees)

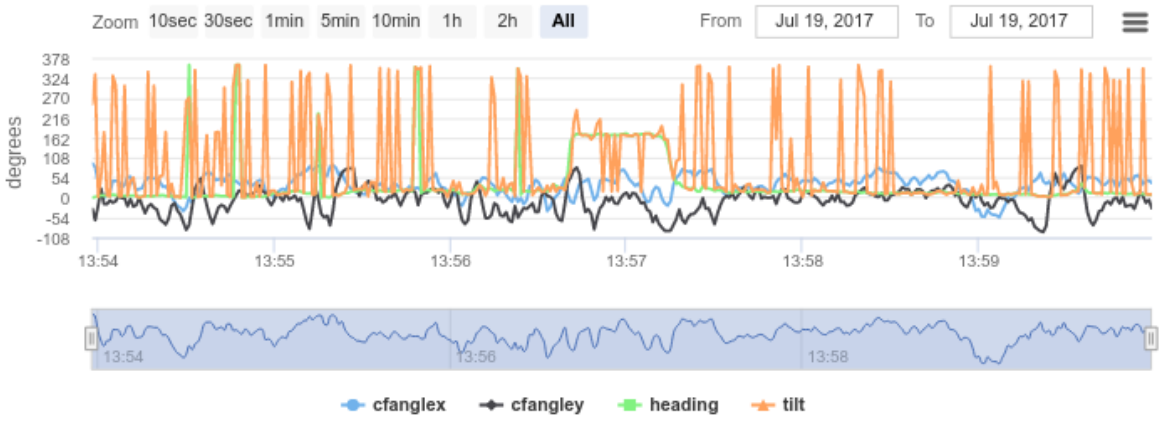


Figure 4.19: Vehicle's behavior chart.

The subtitles identify each line by color and symbol when the mouse hovers them. In this chart our objective was to join all parameters related to the vehicle's behavior. The values calculated by the In-car Unit express angles that are between 0 and 360.

A final adding to this trip viewer, was the weather chart. That graphic, depicted in Figure 4.20, is capable of comparing, for instance, the vehicle's velocity at a specific location, from one trip with precipitation, and another trip in the same area but without any precipitation. This comparison was not the major objective of this page, once we should use another approach to achieve a type of comparison like this.

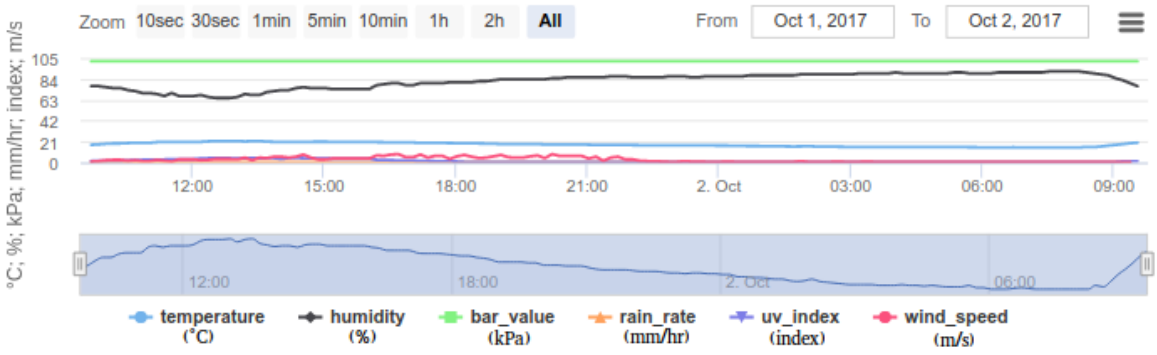


Figure 4.20: Weather's chart.

From all of the possible parameters to display in the Weather's chart, were chosen the following ones:

- Temperature (degrees)
- Humidity (percentage)
- Bar value (kPa)
- Rain Rate (millimeters per hour)
- Ultra-Violet Index (UV)
- Wind Speed (meters per second)

The chosen parameters above belongs to the fundamental weather metrics that are being collected by the backend task. Other metrics as the Temperature Humidity Wind Index (THW) or Solar Radiation, for instance, were not classified as possible parameters to include in the chart.

The disadvantage of joining these parameters is the full visibility of some metrics. The pressure and the humidity values are generally around the one hundred, thus causing a scale changing in the chart, so the rest of the parameters cannot be seen as we want. But this is nothing serious, once we can hide those parameters and the chart automatically re-scale itself. Another way to solve this is using the zooming function.

Another function provided by the dashboard is the possibility to import CSV files that was generated by both Smartphone and In-car Unit. In this page, the user chooses its device on the drop-down menu and then uploads the file. The same output is displayed to the user, a map with the path drawn, and the charts with all information

parsed from the file. Despite showing the weather chart, it is not parsed from the file once it is something that is made only by the backend and then stored on the IoT Platform.

Going back on the website, and leaving the *Devices menu*, we can navigate into the *Weather menu*. This menu splits in three categories, which are: **a)** Weather now **b)** Weather over time and **c)** Today's Highs/Lows.

The first menu will show the most updated weather data. Figure 4.21 displays all the weather data and represents an instant value of each one of the following parameters:

- Temperature (℃)
- Barometer (hPa)
- UV (index)
- Wind Chill (℃)
- Wind direction (cardinal direction)
- Rain Rate (mm/h)
- THW index (℃)
- Solar Rad (W/m^2)
- Storm Total (mm)
- Humidity (%)
- Heat index (℃)
- Dew-point (℃)
- Wind Speed (m/s)

Some of the parameters are usually used on daily routine. Others can be somewhat unknown to the common people, such as the THW index or Solar Radiation. According to [16], the Temperature Humidity Wind Index uses other different metrics (humidity, and temperature) to calculate the apparent temperature, but it also includes the solar radiation influence, and the wind to obtain a real perception of the temperature.

Figure 4.21: Weather now web page.

For the Heat index is used the outside temperature and humidity, to calculate the heat level that is really felt. In this determination of heat index, the humidity is the

key value to provide the result. To determine Dew-point, the same parameters are used. This metric defines the temperature of the air to occur an air saturation[1], thus predicting the formation of dew, frost, and fog [16].

The last of the less known metrics is the Wind Chill, which uses the air temperature and the wind speed. The objective is to discover the wind's temperature felt by taking into account the speed of the wind [16].

The next page is the Weather over time where several charts are displayed as in Figure 4.22. There are a few options on this page, which is to let the user choose the period to be displayed. It can be since one hour to five consecutive days of data.

Despite the period of data to be displayed, the charts show the essential weather's metrics. With the mouse over the charts, it is possible to see a pop-up message with the value and date/time. We also set markers which express the highs/lows from all data in the chart.

Figure 4.22: Weather over time web page.

Lastly, Figure 4.23 displays the highest and lowest values for all possible weather parameters. Although some of the metrics do not have a maximum and a minimum, yet they have at least the highest read value.

The first four metrics, specifically the Temperature, the Dew-point, the Humidity, and the Barometer, they have both maximum and minimum values, and also an associated time. The rest of the parameters show the highest measured value as well as the time, on the current day. At the bottom of Figure 4.23, are depicted the values for

[1] 100% of humidity.

the rain, where is presented the total precipitation for the current day, for the present month, and for all year.

Figure 4.23: Weather Today's Highs/Lows web page.

The dashboard is kind of a final product to the client, it must be interesting, readable, but more important, it should be a work tool for the client analyze the collected information with the potential to complete all requirements of the solution.

4.4.1 DRIVING CHARACTERIZATION

A classification about the driving pattern appears right after the user selects the intended trip, as shown in Figure 4.24. As soon as the website gets the data belonging to the selected trip, the system performs a process to analyze the driver behavior based on one parameter, the *Engine Load*. The determination is very superficial and not so reliable as it should, but it possibly gives an approximated characterization of the driving pattern.

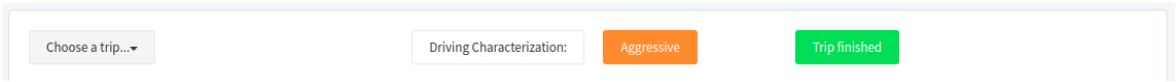


Figure 4.24: Driving pattern characterization.

The method to classify the driving pattern relies on counting the peaks of load on the engine. It holds four levels of counting peaks, which are: **a)** [0%; 35%[**b)** [35%;

50%[**c)** [50%; 80%[and **d)** [80%; 100%]. Where the first corresponds to a *Normal* behavior, the second to a *Moderate* behavior, the third to an *Aggressive* driving and the fourth is the *Racer* where are the most hastened drivers. When no classification is possible to be determined, or if it is a mobile application's trip, the algorithm returns *Unknown*.

For each one, it counts the number of occurrences, and then it transforms them into values of percentage. The counting process works by iterating over each one of the engine load values and then determine in which of the four levels it fits. After this process, it must transform the values into a percentage of the total, to work later with percentages values and not with absolutes, which does not give us any perception of the real engine use, and very important, to become a dynamic process capable of handle any dataset, with any size or duration.

Once it has these percentages, the algorithm proceeds to the phase of behavior characterization. At this point, we needed to discover some threshold values to use in the driver behavior algorithm. Before defining those threshold values was built a characterization tree, depicted in Figure 4.25, to help understanding how to make the system generate an evaluation. The algorithm implements a straightforward interpretation with a low level of filtering. This means that not all percentages are included on the algorithm, so it uses only the highest.

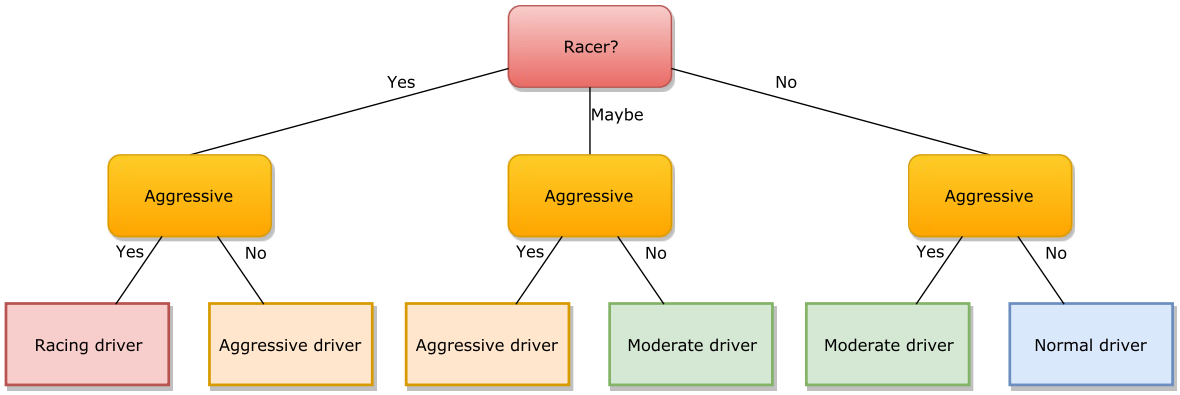


Figure 4.25: Behavior characterization algorithm.

We define the thresholds using just our analysis of some tests made in the field, and they were adopted as reference values. The reference values were obtained using the In-car tracker, and in a path defined by us. We drove through that route using different driving patterns. Thus, we could compare the results, of the percentages of each one the peaks intervals, and we define a set of thresholds to be used by the algorithm.

Table 4.2 is a small list of reference values, but was the best we could get. The letters in the column *Path* represents the starting and stopping points for the route, then in the next column, there is the date and the duration of each path. Finally, follows the

Table 4.2: Our classification for the results of the reference tests.

#	Path	Date/Duration	<35%	35-50%	50-80%	>80%	Pattern
1	A - B	2017-10-19 (12 min)	84%	9%	4%	2%	Normal
2	B - A	2017-10-19 (9 min)	80%	8%	9%	3%	Moderate
3	A - B	2017-10-20 (7 min)	82%	11%	6%	1%	Normal
4	B - A	2017-10-20 (4 min)	72%	12%	11%	5%	Racer
5	C - D	2017-10-20 (7 min)	82%	5%	8%	4%	Aggressive
6	D - C	2017-10-20 (5 min)	74%	5%	11%	10%	Racer

results for each interval in percentages, and then our purposed driving characterization. Were performed some different driving types, from a *normal* driving to a more *racing* driving.

Based on Table 4.2, were defined a set of threshold values for the characterization algorithm. The next results are only for the *80+* column and not for anyone other. As aforementioned, this solution is not the best neither the most accurate, but the values results in:

$$\begin{cases} > 4\%, & \textit{Racer} \text{ and } \textit{Aggressive} \text{ patterns} \\ = 3\%, & \textit{Aggressive} \text{ and } \textit{Moderate} \text{ patterns} \\ < 3\%, & \textit{Moderate} \text{ and } \textit{Normal} \text{ patterns} \end{cases}$$

As shown in the above formula, it always results in a set of two patterns. We went one more level deep, and it was defined a threshold value for the next condition which represents only the *50-80* column. This is easily seen looking into the Figure 4.25, which has one first level, corresponding to the above formula, and then the second level with the same condition defined by the following equation:

$$\begin{cases} > 9\%, & \text{choose the highest pattern from the given set} \\ <= 9\%, & \text{choose the lowest pattern from the given set} \end{cases}$$

Essentially it is a choice between one pattern or the other that was inherited from the top condition (the one given by the first formula).

After the second condition it filters one individual final result, despite being only from the two highest peak's ranges, the result is satisfactory enough taking into account the weak algorithm in use.

CHAPTER 5

Evaluations and Results

Evaluating the tests' output is an essential process of any project. This Section presents an evaluation of the results and experiments under the scope of this dissertation.

5.1 deployment scenario

To proceed with the tests, it must be defined a few things before we start. Firstly, we should have in mind precisely what needs to be collected from all the devices and sources that were implemented by the solution. The following enumeration details those metrics.

Positioning: GPS positioning for all types of devices to track the trips path.

- Latitude *(degrees)*
- Longitude *(degrees)*
- Altitude *(meters)*
- Time *(seconds)*

OBDII: Car data bus can offer key metrics for vehicle characterization. Information about the state of the engine like RPMs or temperature is essential to estimate driving pattern.

- Fuel Level *(%)*
- Speed *(kms/h)*
- RPM (revolutions)
- Engine Load *(%)*
- Coolant Temperature *(℃)*
- Run time *(seconds)*

Vehicle-Specific Power: Considering vehicle behavior as a variable to determine emissions, VSP allows correlating emissions with driver behavior making these metrics valuable.

- VSP value *(kW/ton)*
- VSP mode *(scale of 14 levels)*

Sensing: Additional metrics about the surrounding environment can be useful for stress prediction, like high/low temperatures can be a stress element for the driver.

- Ambient Temperature *(℃)*
- Pressure *(kPa)*

Attitude: The angles allow detecting aggressive curves or movements as accelerations or sharp bends.

- Pitch *(degrees)*
- Roll *(degrees)*
- Heading *(degrees)*
- Tilt Compensated *(degrees)*

Weather: The climate influences the driving behavior, so these metrics offer helpful information to evaluate unusual practices adopted by the drivers.

- Temperature *(℃)*
- Humidity *(%)*
- Barometric Pressure *(kPa)*
- Rain Rate *(mm/hour)*
- UV *(index)*
- Wind Speed *(meters/second)*

Some tests were performed while developing the solution, either to test new features or to begin to see some results of the system. The most straightforward tests were with the mobile application, since it is more user-friendly than the car tracker.

We went numerous times, to the streets to test both devices collecting data. We focused on checking the quality of the data acquisition, in the effect/influence on the GPS signal while driving around the downtown, as well as the proper metrics from all the sensors in use.

The mobile application is the most accessible device to use, once we only need to place the smartphone on a phone holder and then to press the start button to begin logging data, just as explained in Section 4.3.1.

However, there is a bit complicated part on the tracking system. The car tracker needs to be positioned in a secure and fixed spot, while driving, it also requires a power source, which can be a power bank or the car's lighter. The setup on the vehicle is depicted in Figure 5.1.

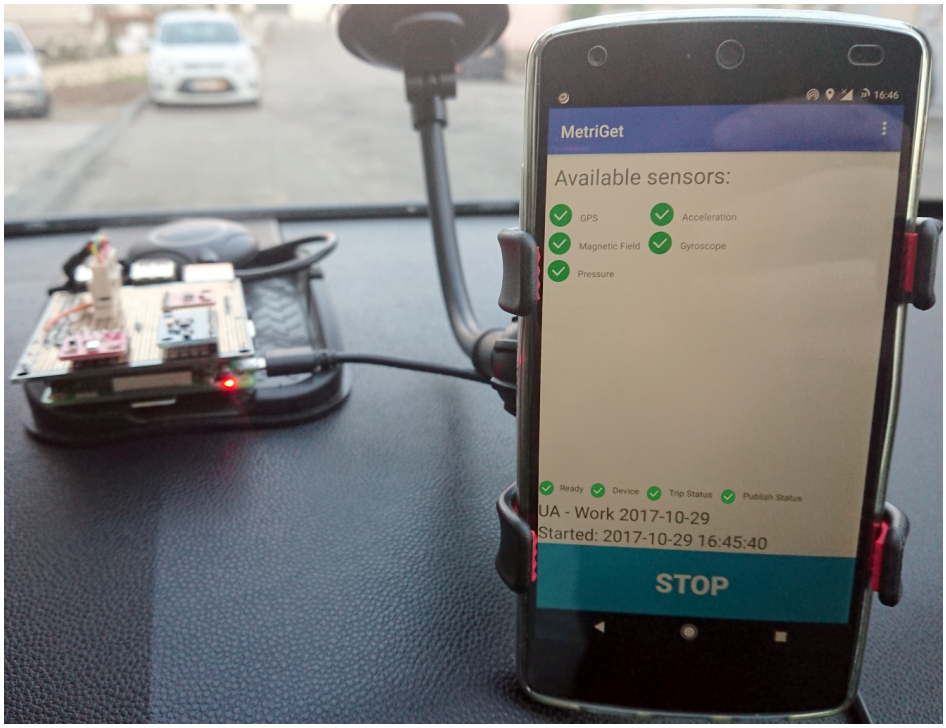


Figure 5.1: Tracking system deployment on a vehicle.

The tests were made mostly in the city of Aveiro, although some of the trips were the city of Oliveira de Azeméis, and also to the city of Coimbra. We have trips since ten minutes till others with almost two hours, and the system behaved well on both cases and without any failure.

5.2 EVALUATION

The evaluation of the deployed scenario makes use of its system, so the tests can be analyzed using the developed platform. Therefore, our approach to inspect and explore the results is to look into the trips analyzer provided by the dashboard.

As explained in Section 4.4, the tracking devices have to create a *Trip* entry on the system before they can start logging any data. Thus, the user will be able to access that particular dataset and make the analysis.

Despite all the other tests which have been made while the development of the system, is possible to specify a route to use as a reference path for the tests. In the example from the University of Aveiro to Cacia Park, Rua do Progresso, in Aveiro, which are around 8 kilometers of distance.

The highlighted route on Figure 5.2 defines the traveled path. There are alternatives marked on the image but we will not consider for our use case. From the Google Maps

website, the best route from one point to another can be easily obtained, including the traffic information (for instance, the lines in red, identified by number 1, marks a congestion spot).

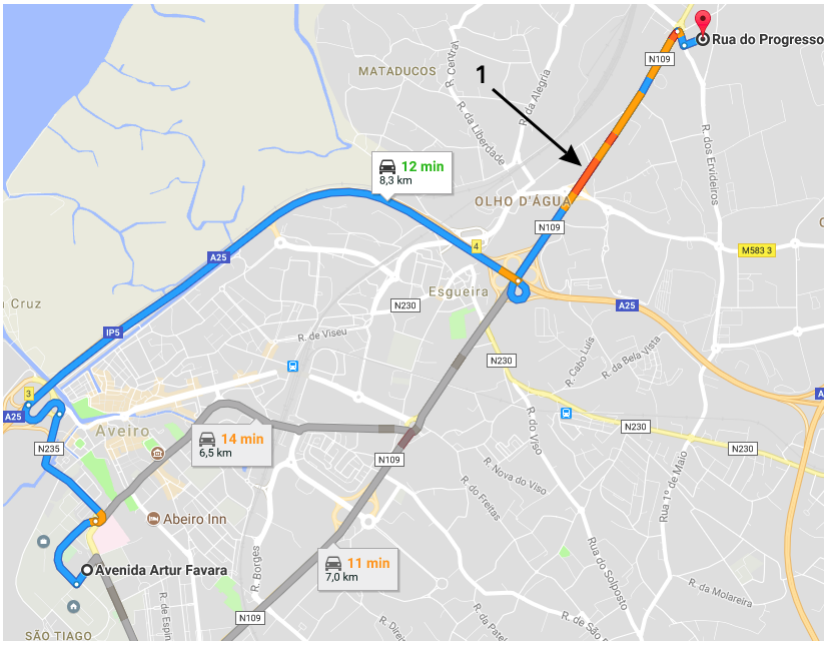


Figure 5.2: Reference path for the tests from Google Maps.

An overall output screen from the tests can be seen in Appendix C. It is an extensive output, but the user is able to see, in the dashboard, each chart with considerable detail, including zoom and even isolating specific metrics hiding the others.

All data was obtained simultaneously by both devices, and thus we can observe more information from multiple data sources. Therefore, there is the advantage of comparing the output data between the different devices. For the driving pattern determination, it will only be characterized for the In-car trips. Appendix C.1 presents the result of the driving pattern evaluation, as detailed in Section 4.4.1.

Appendices C.2 and C.3 were taken from the In-car device and mobile application, respectively. They correspond to the same path at the same time. The perception of the precision differences between both GPS signals is somewhat evident which is a natural aspect when dealing with GPS receivers so distinct in price, precision, capacity or purpose.

Charts in appendices C.4 and C.5 displays the measured attitude from the vehicle over the time. The methods used to perform the attitude calculation are different for each device, as explained in Section 4.3.2 for the car tracker and in Section 4.3.1 for the mobile application. Both techniques give a bit different values, but we can see that the same distinct values generated by bigger changes in the attitude are perceptible either in the In-car chart as in the mobile application chart.

The set of charts belonging to the appendix C.6 are acquired only from the In-car output. These data are mostly obtained from the OBDII data and from the calculation of the VSP variable. The subtitles provided by each graphic are identified by color and name and can be deactivated reducing the analysis complexity. The first of the three charts houses most of the data including the metrics from OBDII as well as from the ambient sensing like the temperature and pressure inside the vehicle's habitat.

The system is capable of providing high detail of a driver's travel metrics. Parameters that might be the most suitable for the purpose. Moreover, with the possibility to add even more metrics without changing the system architecture. A vast future work is in sight, in order to provide new features, but mostly, to improve the existent infrastructure.

5.3 RESULTS

In this section, is presenting an interpretation of the driving characterization results and some comparisons. Then we will take a look at the influence of congestion streets in the obtained parameters, for instance, what changes in speed and in the revolutions per minute. Finally, it is shown an overview of the detection of gear shifts and its impact on vehicle's emissions.

5.3.1 DRIVING CHARACTERIZATION

The driver behavior is a crucial detail on many aspects of traffic congestion. Besides the risk of accidents, an aggressive driver can be more stressful to the other drivers, and for sure, its driving method increases the amount of CO2 emissions.

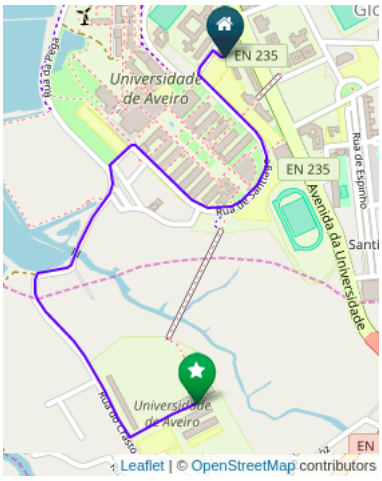


Figure 5.3: GPS path from point A to B.

Figures 5.4, 5.5, 5.6, and 5.7 are resulting from a set of trips made from the University Residences to the Crasto's canteen on October 21 and 22 of 2017, depicted in Figure 5.3, and they are identified as point A and B, respectively. They are shown ordered by date and first from A to B and then B to A points.

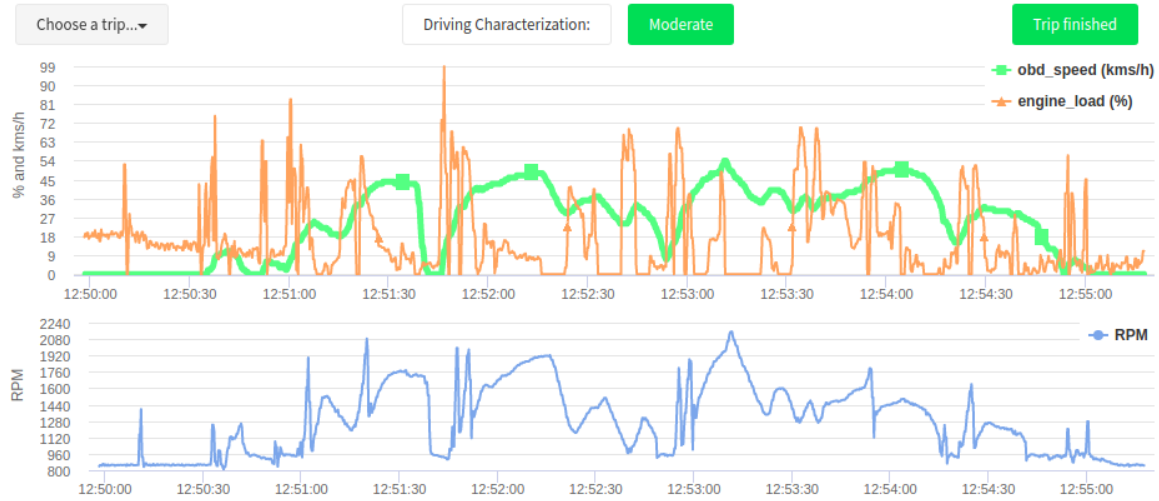


Figure 5.4: Driving pattern for the path from point A to B on October, 21.

Figure 5.5: Driving pattern for the path from point B to A on October, 21.

Each Figure depicts a different driving pattern. This was compelled by us, to express the differences between each one of the driving patterns defined in the Section 4.4.1. We drive with behaviors since a racing mode to a smooth and less polluting way. Thereby, we were able to get all the possible behavior outputs, and in our perspective, the algorithm is working good.

The differences between the driving modes are perceptible looking into the charts. As we can see in the subtitles, the lines in orange (thinner line) are the *Engine Load*,

the lines in green (thicker line) are the *Speed*, and the lines in blue on the second chart are the *Revolutions per Minute*.

The correlation of each line is distinguishable, that is every time we see a peak in the RPMs (the line in blue), it triggers two things, one is the engine load that suffers a peak too, which leads to the second event, increasing the vehicle's speed.

In some cases, we can watch the speed's line (in green and thicker), with a level relatively high but with the line of the engine load considerably low. We can learn from this fact, that the peaks in engine load are triggered when the driver suddenly accelerates, raising the RPMs very quickly.

Even with high velocity and RPMs, it is possible to watch a low level of load on the engine. But when it comes to increasing the acceleration abruptly, the impact on the engine load is highly increased.

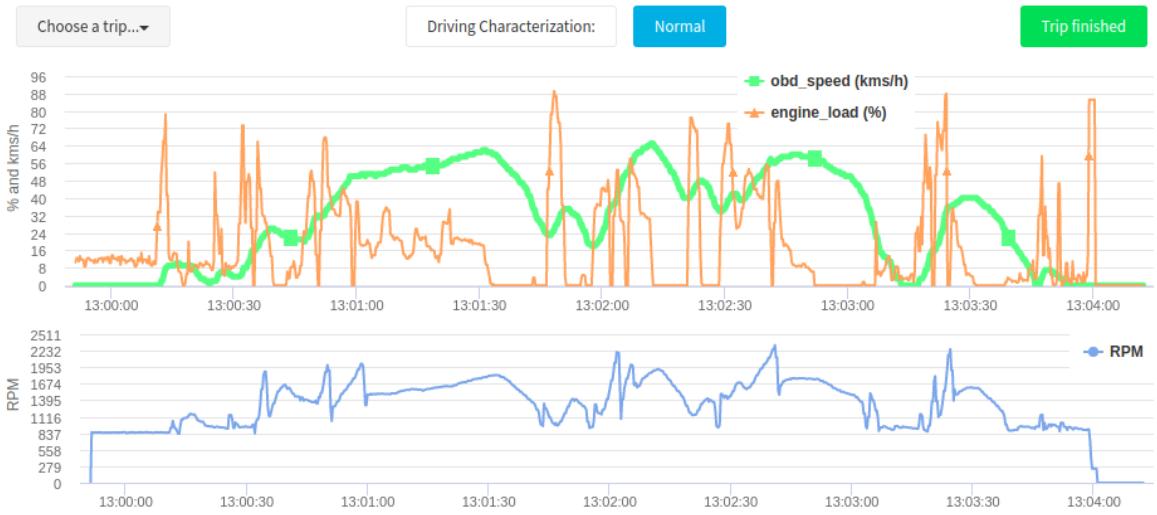


Figure 5.6: Driving pattern for the path from point A to B on October, 22.

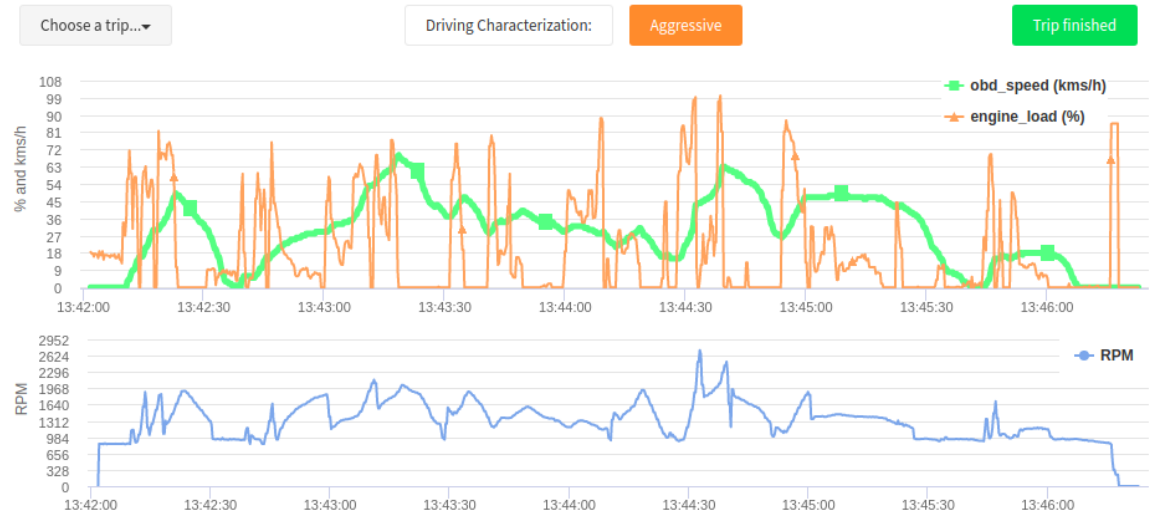


Figure 5.7: Driving pattern for the path from point B to A on October, 22.

Comparing the driving pattern in Figure 5.5 with the adopted driving pattern in Figure 5.6 is evident that the first one practices a way more risky driving method than the second. We can watch the line in blue more sharpen with short intervals between peaks of the RPMs, as well as the quick raising of the speed.

For a final note on this section, we are safe to claim the success of the driving characterization algorithm. For us, as human beings, it is quite easy to see these differences between a pattern to the other and with the help of colored lines, but to the computers is not so easy, and it must be designed thinking for all possible scenarios, error proof, and many other circumstances.

5.3.2 CONGESTION DETECTION

Alternatively to the previous Section, where the system provides an analysis of the driver behavior, this Section will explain a way to detect parts of one trip where probably there were traffic jams. A manual study that can be done by any person watching a determined trip through the dashboard.

Like in Section 5.3.1, we are again looking into the trip's outputs, but now we will go to areas where usually we take an expected time to do it, but due to several externalities, we might take much more time. Such externalities can be many, for instance:

- Rush hour
- Accidents
- Roads in bad conditions
- In some cities of the world, the smog
- Weather conditions
- Vehicle's failures
- Inexperienced drivers

The most common congestion factor is most likely the first of the list, the rush hour. Every day, for around the world during the rush hours, there is no way to escape the traffic jam, unless you are living in a rural place.

Weather conditions compromise the traffic flow more than we thought. As drivers tend to drive with precaution when raining or when there is fog.

Our case study will be about the most common congestion factor, the rush hour. We used a fragment of the trip between the University of Aveiro and the Cacia Park, Rua do Progresso, in Aveiro, identified in Section 5.1. That part belongs to the most congested road of the trip. Figure 5.8 displays that section where the traffic flow has a significant difference at different hours.

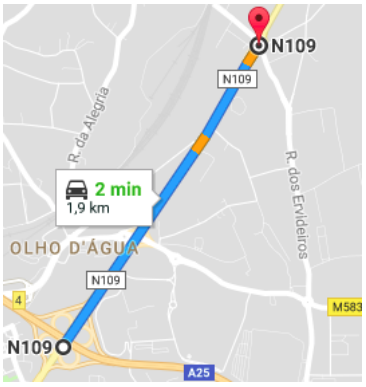
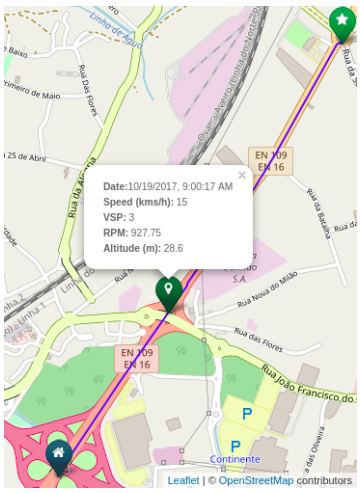


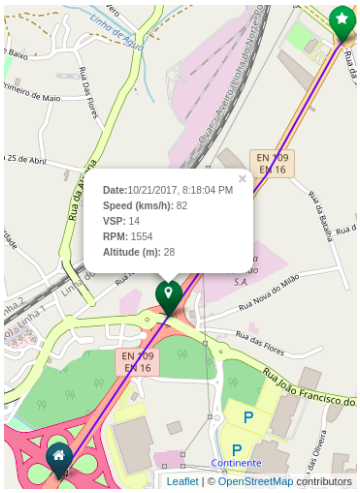
Figure 5.8: Route to be analyzed.

Sub-figures 5.9a and 5.9b are showing the most congested street of the trip between the University of Aveiro and Cacia Park, Rua do Progresso. Figure 5.9a presents a trip at the rush hour, while Figure 5.9b shows the opposite.

It is possible to see some information on the map, shown by a pop-up. Such information is related to the point on the map, and it contains some of the metrics that are also available on the other charts which belongs to that trip.



(a) Traffic flow during rush hour.



(b) Traffic flow outside the rush hour.

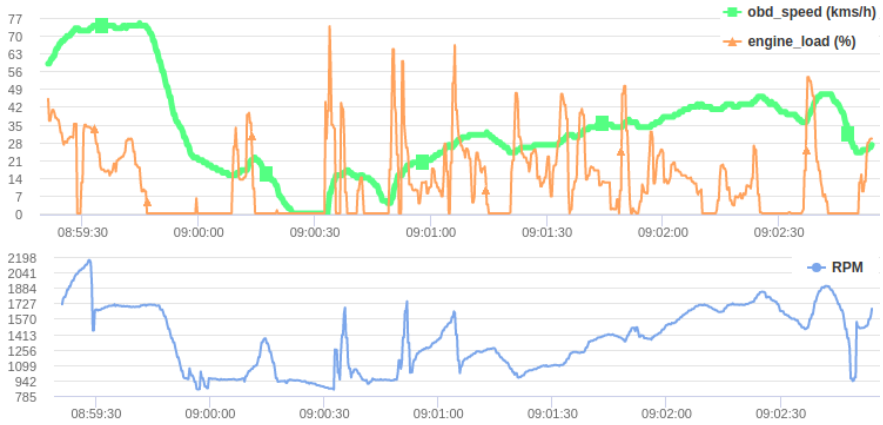
Figure 5.9: Comparison of output charts for both trips.

The first thing that can tell us something about the traffic flow is the duration time needed to travel it. Despite we can not see that in the images, it is possible to watch it on the charts depicted in Figure 5.10. Looking closely at both charts, they depict distinct timelines, while sub-figure 5.10a has a duration of about three minutes, sub-figure 5.10b depicts a duration about one minute. The charts illustrate that, for the same path, were practiced different speeds and different elapsed times.

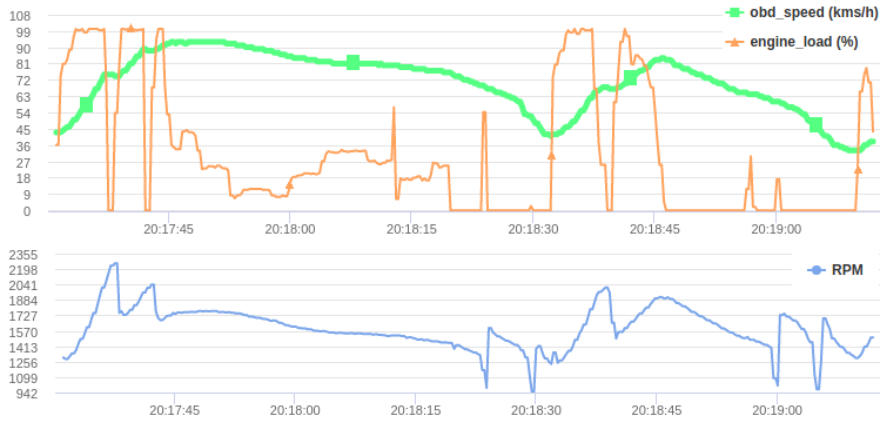
Comparing both charts, including the RPMs, the first thing that pops out, between them, is the difference in the number of peaks. Sub-figure 5.10a presents low speeds

(sometimes stopped), and a significant amount of engine load peaks. We quickly find out what was the problem, a traffic jam. The high number of engine load peaks with low speeds means that the driver is continuously accelerating and braking excessively, over and over.

Meanwhile, Sub-figure 5.10b displays higher speeds, much less engine load peaks, as well as way more constant RPMs.



(a) During rush hour.



(b) Outside the rush hour.

Figure 5.10: Comparison of output charts for both trips.

Concluding this subsection, the congestion problem will always be a problem for the traffic management systems. For now, the solution is, in some cases, avoid the traffic jams using alternative routes. Other solutions can be the use of public transportation systems, which can leave us out of the stress of driving during the rush hours, and contributing to the decreasing of ambient impact.

5.3.3 DRIVING BEHAVIOR AND IMPACT ON VSP

Nowadays the environment concerns, are a current aspect in the political questions. The effects of the global warming are upon us. Many efforts are being implemented to solve this problem, as well as to prevent and teach the society to be more aware of the ambient aspects and how to avoid the wasting and excesses practices.

In this Section, we will approach this particular case, which is an emergent aspect to be considered when dealing with traffic management. Congested roads concentrate high levels of pollutants, so an ATMS might provide, for instance, alternative routes to let the drivers avoid, not only the traffic jams but also the vehicles' emissions.

In Section 4.3.2, the VSP parameter is assessed, where is combined an aspect related to the driver behavior and the ambient factor. Such parameter, let us determine and identify the impacts of the driver behavior in the vehicle's emissions.

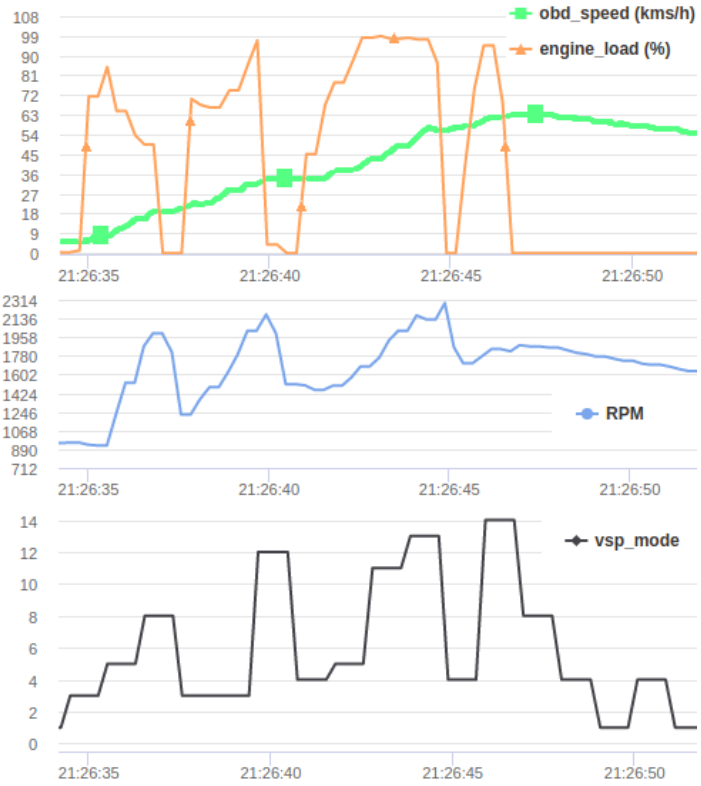


Figure 5.11: Driver behavior and its impact on emissions (VSP).

The high correlation of the vehicle's VSP with its emissions can be visible on the analysis of the dashboard output. Figure 5.11 displays exactly that fact. It is a joint of three different charts, but all in the same time window and belonging to the same trip.

The first chart, composed of two lines, displays four engine load peaks and consequently the speed increasing. Along with that, we have the second chart, with the RPMs, showing the same scenario. As the driver demands more speed by pressing the throttle, an RPM peak appear, until the driver executes a gear shift, and another peak

comes up again related to the engine load and speed. Hence, we can detect variations like these, and determine, for instance, when the driver makes more gear shifts and thus, where he could reduce the emissions impact with fewer aggressive behaviors.

The last chart shows the VSP mode derived from the other graphs. The relation of the driver behavior with the emissions impact can be easily seen on these three charts. More aggressive drivers will see this pattern more times, and for sure they will notice in fuel consumption.

CHAPTER 6

Conclusions

The increase in road traffic led to the creation of Advanced Traffic Management Systems. The implementation of the solution presented in this dissertation uses several state-of-the-art techniques offered with the exponential growth in the topic of Intelligent Transportation Systems.

The work performed during this dissertation let us learn, in depth, the IoT concept, as well as the traffic congestion problem and its ambient impact. The road transports will always be a key factor for the climate change. Despite the efforts to develop ambient friend engines, new hybrid motors, and the growth of electrical engines, the expected contribution of oil products in 2050 will be 50% [19].

Therefore, our solution, in the scope of @CRUiSE project, aims to a more efficient use of existing infrastructures, and the potential for emissions reduction through the Intelligent Transportation Systems. The main tasks defined in the beginning, were in first place, the development of a Floating Car Data using a mobile phone, then an In-car tracking device. Since we got these devices the next task was to built a system capable of providing and to combine different sources of data on a single platform, and then each network user could take advantage of safer roads, find out where he takes more time to its destination, or where are the most congested streets in the city.

The collected data and the platform will be available for use and further processing to other researchers. Moreover, as an integral part of @CRUiSE project, the platform, developed devices, and acquired data will be available.

Everything displayed and collected under the scope of this dissertation is available online for consulting at `http://cruise.aws.atnog.av.it.pt`.

6.1 FUTURE WORK

A vast future work is in sight. Such work, either to add new and pertinent features or to improve current ones, capable of bringing great improvements to the overall solution. The following list describes some improvements that might be a great enhancement to the system:

a) Add support to connect the vehicle data bus through OBDII to the mobile application.

b) Providing weather information for the closest position of the logging device in use.

c) Addition of different data sources, like population (crowdsourcing).

d) Implement a better behavior analysis.

e) Process data aggregates to rely on large amounts of data.

f) Link-based analysis, comparing data by location instead of the concept of trips.

There are numerous improvements, either adding new features to the system either to enhance and improve existing ones. Topic **a)** will allow having a mobile application capable of providing similar data as the In-car tracker. Topic **b)** aims to provide information about weather state but in the form of following the positioning of the tracker in use. Crowdsourcing, enumerated in topic **c)**, might provide a good improvement in the data sources domain, yet represents a challenging method to develop. To enhance the behavior algorithm, in topic **d)**, can be applied machine learning techniques which raises the algorithm to a different level of performance and reliability. Topic **e)** refers the large amounts of data, that can help to reduce error. Topic **f)** aims to add fixed points to acquire data in separated links of the city, then provide comparisons between different trips and, for instance, to joint information about the state of the road.

APPENDIX A

Nginx

```
# backend_nginx.conf
upstream wsgi_django {
    server unix:///tmp/wsgi_django.sock;
}
upstream storage {
    server storage-scot.vm.atnog.av.it.pt fail_timeout=0;
}
upstream supervisord {
    server localhost:9001 fail_timeout=0;
}

# configuration of the server
server {
    listen          80; # listening port
    server_name        cruise.aws.atnog.av.it.pt; # domain
    charset         utf-8;

    client_max_body_size 100M; # max upload size

    # Django static files
    location /media  {
        alias /home/cruise/backend/frontend;
    }
    location /static {
        alias /home/cruise/backend/static;
    }
    location /app/ {
        alias /home/cruise/backend/frontend/app/;
    }
    location /server/ {
```

```
        alias /home/cruise/backend/frontend/server/;
    }
    location /vendor/ {
        alias /home/cruise/backend/frontend/vendor/;
    }

    # Send all non-media requests to the Django server.
    location / {
        uwsgi_pass  wsgi_django;
        include     /etc/nginx/uwsgi_params;
    }

    include /etc/nginx/site-includes/*;
}
```

APPENDIX B

APIs

Table B.1: Backend root's API.

method	**route**	**description**
GET	/	Root path for website
GET	/admin/	Admin site's API
GET	/api/docs/	Swagger documentation's API
GET	/api/auth/	Prefix for authentication's API
GET	/api/devices/	Prefix for devices' API
GET	/api/proxy/(?P<path>.*)$	Path for proxy requests
GET	/app/(.*)$	Application static files
GET	/server/(.*)$	Server static files
GET	/vendor/(.*)$	3rd party static files

Table B.2: Authentication's API.

method	**route**	**description**
POST	/api-token-auth/	Authenticates and returns a JSON Web Token
POST	/login/	Authenticates the user in the website
POST	/logout/	Logout the user from the website
POST	/register/	Register new user

Table B.3: Devices' API.

method	route	description
GET	/	Allows to get a device by given parameter
GET	/all/	Returns a list of all devices
GET	/types/	Returns all possible types of sensors
POST	/info/(?P[a-zA-Z0-9]+)/$	Get Device info by given UUID
GET	/trips/(?P[a-zA-Z0-9]+)/$	Get Device trips by given UUID
POST	/trips/(?P[a-zA-Z0-9]+)/$	Register/update trip from given device id
POST	/sensors/(?P[a-zA-Z0-9]+)/$	Get Device Sensors by given UUID

APPENDIX C

Output Screens

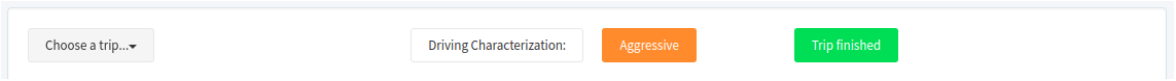


Figure C.1: Driving pattern characterization.

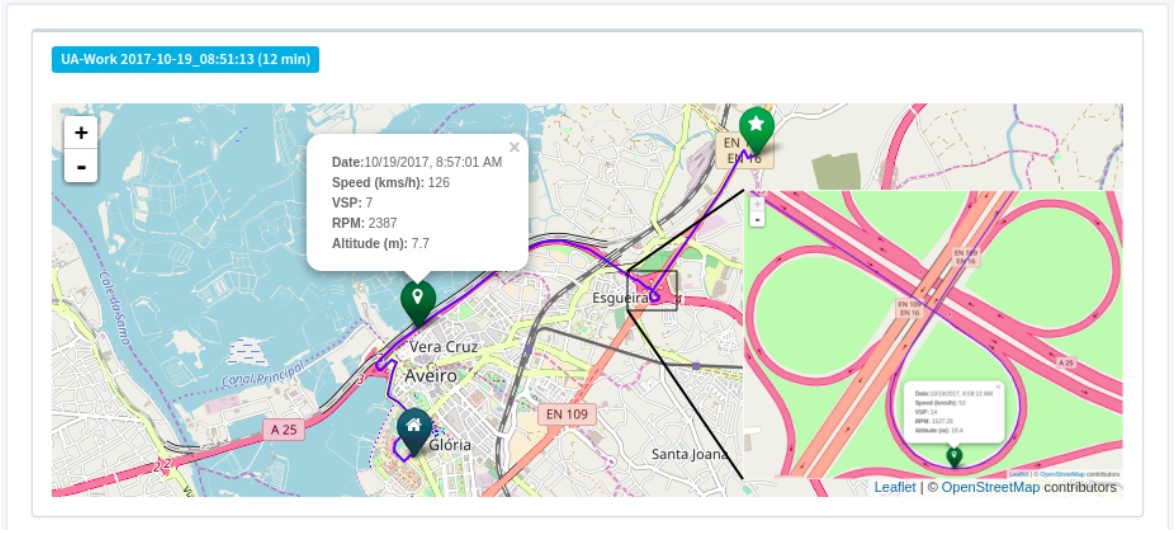


Figure C.2: GPS path from the car tracker.

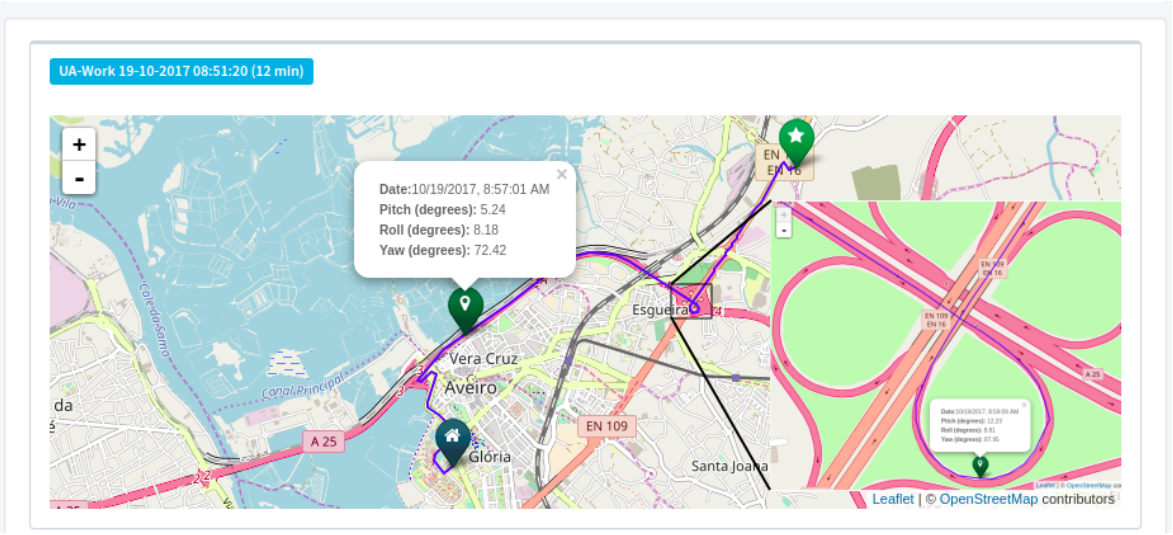


Figure C.3: GPS path from the mobile application.

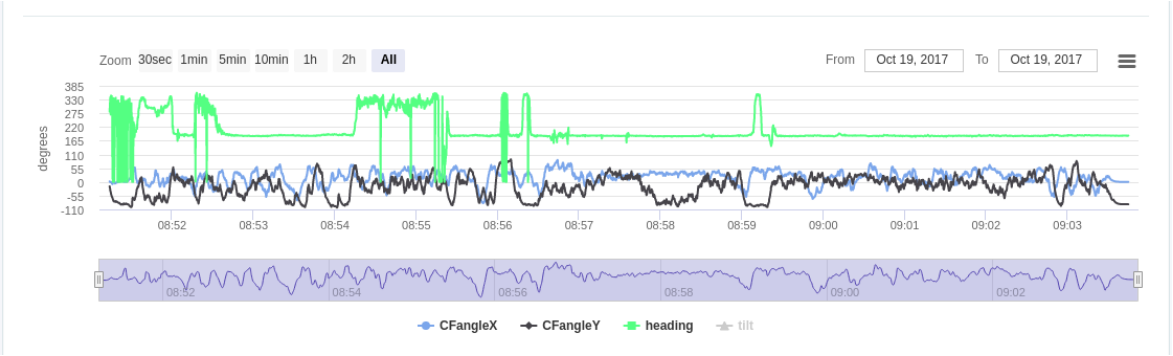


Figure C.4: Vehicle attitude measured by car tracker over the time.

Figure C.5: Vehicle attitude measured by the mobile application over the time.

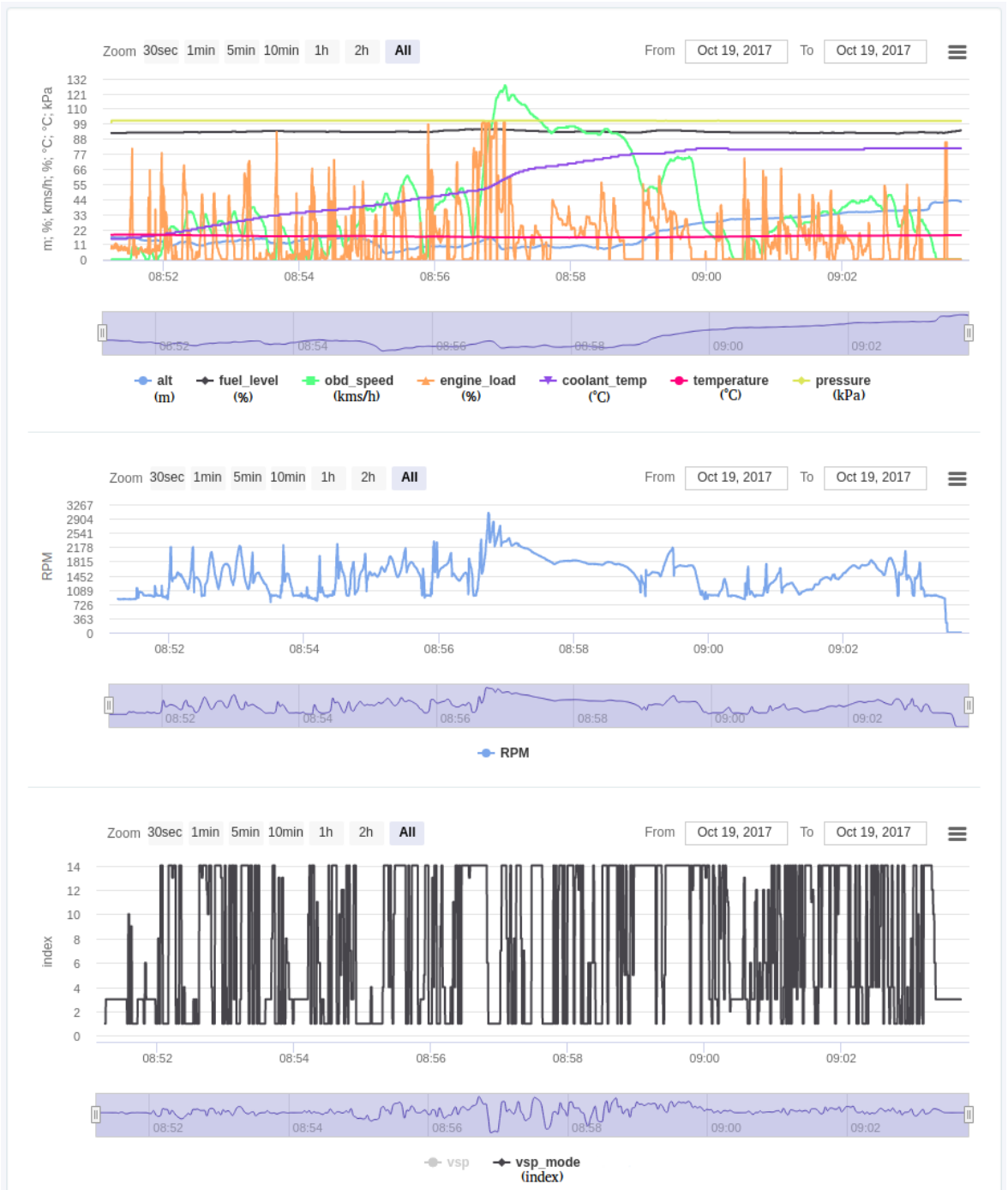


Figure C.6: Overall output about OBDII metrics and the VSP calculation.

References

[1] T. Adame, A. Bel, B. Bellalta, J. Barcelo, and M. Oliver, "Ieee 802.11ah: the wifi approach for m2m communications", *IEEE Wireless Communications*, vol. 21, no. 6, pp. 144–152, 2014, ISSN: 1536-1284. DOI: 10.1109/MWC.2014.7000982. [Online]. Available: http://ieeexplore.ieee.org/document/7000982/.

[2] D. Airehrour, J. Gutierrez, and S. K. Ray, "Secure routing for internet of things: a survey", *Journal of Network and Computer Applications*, vol. 66, pp. 198–213, 2016. DOI: 10.1016/j.jnca.2016.03.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516300133.

[3] M. Antunes, J. P. Barraca, D. Gomes, P. Oliveira, and R. L. Aguiar, "Smart cloud of things: an evolved iot platform for telco providers", *Journal of Ambientcom*, vol. 1, no. 1, pp. 1–24, 2015. DOI: 10.13052/AMBIENTCOM2246-3410.111. [Online]. Available: http://riverpublishers.com/journal/journal_articles/RP_Journal_2246-3410_111.pdf.

[4] O. Armas, R. García-Contreras, and Á. Ramos, "Impact of alternative fuels on performance and pollutant emissions of a light duty engine tested under the new european driving cycle", *Applied Energy*, vol. 107, pp. 183–190, Jul. 2013, ISSN: 0306-2619. DOI: 10.1016/J.APENERGY.2013.01.064. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261913000755.

[5] H. Arslan, Ed., *Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems*. Dordrecht: Springer Netherlands, 2007, ISBN: 978-1-4020-5541-6. DOI: 10.1007/978-1-4020-5542-3. [Online]. Available: http://link.springer.com/10.1007/978-1-4020-5542-3.

[6] F. Azzola, *Mqtt protocol tutorial: step by step guide*, 2016. [Online]. Available: https://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html.

[7] C. Bachmann, M. J. Roorda, B. Abdulhai, and B. Moshiri, "Fusing a bluetooth traffic monitoring system with loop detector data for improved freeway traffic speed estimation", *Journal of Intelligent Transportation Systems*, vol. 17, no. 2, pp. 152–164, Apr. 2013. DOI: 10.1080/15472450.2012.696449. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/15472450.2012.696449.

[8] G. Baldini, T. Sturman, A. R. Biswas, R. Leschhorn, G. Godor, and M. Street, "Security aspects in software defined radio and cognitive radio networks: a survey and a way ahead", *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 355–379, 2012. DOI: 10.1109/SURV.2011.032511.00097. [Online]. Available: http://ieeexplore.ieee.org/document/5742780/.

[9] J. Bandeira, T. G. Almeida, A. J. Khattak, N. M. Rouphail, and M. C. Coelho, "Generating emissions information for route selection: experimental monitoring and routes characterization", *Journal of Intelligent Transportation Systems*, vol. 17, no. 1, pp. 3–17, Jan. 2013, ISSN: 1547-2450. DOI: 10.1080/15472450.2012.706197. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/15472450.2012.706197.

[10] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures", in *Proceedings of the 14th ACM international conference on Mobile computing*

and networking - MobiCom '08, New York, New York, USA: ACM Press, 2008, p. 116, ISBN: 9781605580968. DOI: 10.1145/1409944.1409959. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1409944.1409959.

[11] J. Burki, F. Malik, and M. Mushtaq, "Gsm downlink protocol analysis and decoding using open-source hardware and software", in *2013 2nd National Conference on Information Assurance (NCIA)*, IEEE, Dec. 2013, pp. 39–46, ISBN: 978-1-4799-1288-9. DOI: 10.1109/NCIA.2013.6725322. [Online]. Available: http://ieeexplore.ieee.org/document/6725322/.

[12] M. Byłak and D. Laskowski, "Assessment of network coding mechanism for the network protocol stack 802.15.4/6lowpan", in *New Results in Dependability and Computer Systems*, Springer, Heidelberg, 2013, pp. 75–82. DOI: 10.1007/978-3-319-00945-2_7. [Online]. Available: http://link.springer.com/10.1007/978-3-319-00945-2_7.

[13] L. Calderoni, D. Maio, and S. Rovis, "Deploying a network of smart cameras for traffic monitoring on a "city kernel"", *Expert Systems with Applications*, vol. 41, no. 2, pp. 502–507, Feb. 2014, ISSN: 0957-4174. DOI: 10.1016/J.ESWA.2013.07.076. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417413005630.

[14] M. Chen, S. Mao, and Y. Liu, "Big data: a survey", *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, Apr. 2014, ISSN: 1383-469X. DOI: 10.1007/s11036-013-0489-0. [Online]. Available: http://link.springer.com/10.1007/s11036-013-0489-0.

[15] Y. Chen and T. Kunz, "Performance evaluation of iot protocols under a constrained wireless access network", 2016. [Online]. Available: http://ieeexplore.ieee.org/ielx7/7492633/7496590/07496622.pdf?tp=&arnumber=7496622&isnumber=7496590.

[16] Climetua, "Derived variables in davis weather products", 2006. [Online]. Available: http://climetua.fis.ua.pt/legacy/main/current_monitor/WLinkExpVar.pdf.

[17] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "Evaluation of constrained application protocol for wireless sensor networks", in *2011 18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, IEEE, Oct. 2011, pp. 1–6, ISBN: 978-1-4577-1264-7. DOI: 10.1109/LANMAN.2011.6076934. [Online]. Available: http://ieeexplore.ieee.org/document/6076934/.

[18] S. Consortium, *Sqlite*. [Online]. Available: https://www.sqlite.org/about.html.

[19] CRUiSE, *@cruise*, 2015. [Online]. Available: https://project-cruise.weebly.com/about.html.

[20] F. Cuomo, A. Abbagnale, and E. Cipollone, "Cross-layer network formation for energy-efficient ieee 802.15.4/zigbee wireless sensor networks", *Ad Hoc Networks*, vol. 11, no. 2, pp. 672–686, 2013, ISSN: 1570-8705. DOI: 10.1016/J.ADHOC.2011.11.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870511002137.

[21] E. Dahlman, S. Parkvall, and J. Sköld, *4G LTE/LTE-Advanced for mobile broadband*. Elsevier/Academic Press, 2013, p. 431, ISBN: 012385489X. [Online]. Available: http://www.sciencedirect.com/science/book/9780123854896.

[22] L. Daniel, M. Kojo, and M. Latvala, "Experimental evaluation of the coap, http and spdy transport services for internet of things", in, Springer, Cham, 2014, pp. 111–123. DOI: 10.1007/978-3-319-11692-1_10. [Online]. Available: http://link.springer.com/10.1007/978-3-319-11692-1_10.

[23] P. Deville, C. Linard, S. Martin, M. Gilbert, F. R. Stevens, A. E. Gaughan, V. D. Blondel, and A. J. Tatem, "Dynamic population mapping using mobile phone data.", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 45, pp. 15 888–93, Nov. 2014. DOI: 10.1073/pnas.1408439111. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/25349388.

[24] C. Doukas, L. Capra, F. Antonelli, E. Jaupaj, A. Tamilin, and I. Carreras, "Providing generic support for iot and m2m for mobile devices", in *The 2015 IEEE RIVF International Conference on Computing & Communication Technologies - Research, Innovation, and Vision for Future (RIVF)*, IEEE, Jan. 2015, pp. 192–197, ISBN: 978-1-4799-8043-7. DOI: 10.1109/RIVF.2015.7049898. [Online]. Available: http://ieeexplore.ieee.org/document/7049898/.

[25] J. Engelbrecht, M. J. Booysen, F. J. Bruwer, and G.-J. van Rooyen, "Survey of smartphone-based sensing in vehicles for intelligent transportation system applications", *IET Intelligent Transport Systems*, vol. 9, no. 10, pp. 924–935, Dec. 2015. DOI: 10.1049/iet-its.2014.0248. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/iet-its.2014.0248.

[26] R. Faragher, "Ieee signal processing magazine [128] understanding the basis of the kalman filter via a simple and intuitive derivation", 2012. DOI: 10.1109/MSP.2012.2203621. [Online]. Available: https://www.cl.cam.ac.uk/%7B~%7Drmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf.

[27] M. Franceschinis, C. Pastrone, M. A. Spirito, and C. Borean, "On the performance of zigbee pro and zigbee ip in ieee 802.15.4 networks", in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, Oct. 2013, pp. 83–88, ISBN: 978-1-4799-0428-0. DOI: 10.1109/WiMOB.2013.6673344. [Online]. Available: http://ieeexplore.ieee.org/document/6673344/.

[28] Y. Gao, T. Chen, M. Li, H. Feng, and B. Chen, "Fuel consumption model for trucks based on vehicle specific power", 2016. [Online]. Available: http://docs.trb.org/prp/17-03959.pdf.

[29] A. Goel, N. Bansal, and S. Gupta, "Comparison of different web servers", *Imperial Journal of Interdisciplinary Research*, vol. 2, no. 12, pp. 2454–1362, 2016. [Online]. Available: http://www.onlinejournal.in.

[30] Google, *Distributions | android developers*, 2017. [Online]. Available: https://developer.android.com/about/dashboards/index.html.

[31] ——, *Introduction to android | android developers*, 2017. [Online]. Available: https://developer.android.com/guide/index.html.

[32] ——, *Sensormanager | android developers*, 2017. [Online]. Available: https://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_DELAY_FASTEST.

[33] O. Hersent, D. Boswarthick, and O. Elloumi, *The Internet of Things: Key Applications and Protocols*. Chichester, UK: John Wiley & Sons, Ltd, Dec. 2011, ISBN: 9781119958352. DOI: 10.1002/9781119958352. [Online]. Available: http://doi.wiley.com/10.1002/9781119958352.

[34] T. Higuchi, H. Yamaguchi, and T. Higashino, "Mobile devices as an infrastructure: a survey of opportunistic sensing technology", *Journal of Information Processing*, vol. 23, no. 2, pp. 94–104, 2015. DOI: 10.2197/ipsjjip.23.94. [Online]. Available: https://www.jstage.jst.go.jp/article/ipsjjip/23/2/23_94/_pdf.

[35] J. Hu, H. C. Frey, and S. S. Washburn, "Comparison of vehicle-specific fuel use and emissions models based on externally and internally observable activity data", *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2570, pp. 30–38, Jan. 2016, ISSN: 0361-1981. DOI: 10.3141/2570-04. [Online]. Available: http://trrjournalonline.trb.org/doi/10.3141/2570-04.

[36] J. Hu, W. Cao, J. Luo, and X. Yu, "Dynamic modeling of urban population travel behavior based on data fusion of mobile phone positioning data and fcd", *17th International Conference on Geoinformatics*, 2009. DOI: 10.1109/GEOINFORMATICS.2009.5293222. [Online]. Available: http://ieeexplore.ieee.org/document/5293222/.

[37] J. L. Jiménez, P. Mcclintock, G. J. Mcrae, D. D. Nelson, and M. S. Zahniser, "Vehicle specific power: a useful parameter for remote sensing and emission studies", *9th CRC On-Road Vehicle Emissions Workshop*, 1999. [Online]. Available: http://cires1.colorado.edu/jimenez/Papers/Jimenez_VSP_9thCRC_99_final.pdf.

[38] 1.-. Jiménez-Palacios, José Luis, "Understanding and quantifying motor vehicle emissions with vehicle specific power and tildas remote sensing", 1999. [Online]. Available: https://dspace.mit.edu/handle/1721.1/44505#files-area.

[39] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform", in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Oct. 2011, pp. 1609–1615, ISBN: 978-1-4577-2197-7. DOI: 10.1109/ITSC.2011.6083078. [Online]. Available: http://ieeexplore.ieee.org/document/6083078/.

[40] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections", *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 108–118, Jun. 2000, ISSN: 15249050. DOI: 10.1109/6979.880968. [Online]. Available: http://ieeexplore.ieee.org/document/880968/.

[41] K. Kanistras, G. Martins, M. J. Rutherford, and K. P. Valavanis, "A survey of unmanned aerial vehicles (uavs) for traffic monitoring", in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, May 2013, pp. 221–234, ISBN: 978-1-4799-0817-2. DOI: 10.1109/ICUAS.2013.6564694. [Online]. Available: http://ieeexplore.ieee.org/document/6564694/.

[42] O. Karabasoglu and J. Michalek, "Influence of driving patterns on life cycle cost and emissions of hybrid and plug-in electric vehicle powertrains", *Energy Policy*, vol. 60, pp. 445–461, Sep. 2013, ISSN: 0301-4215. DOI: 10.1016/J.ENPOL.2013.03.047. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0301421513002255.

[43] W. Kellerer and H.-J. Vogel, "A communication gateway for infrastructure-independent 4g wireless access", *IEEE Communications Magazine*, vol. 40, no. 3, pp. 126–131, Mar. 2002. DOI: 10.1109/35.989771. [Online]. Available: http://ieeexplore.ieee.org/document/989771/.

[44] M. Khanafer, M. Guennoun, and H. T. Mouftah, "A survey of beacon-enabled ieee 802.15.4 mac protocols in wireless sensor networks", *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 856–876, 2014, ISSN: 1553-877X. DOI: 10.1109/SURV.2013.112613.00094. [Online]. Available: http://ieeexplore.ieee.org/document/6687312/.

[45] J. Korhonen, *Introduction to 4G mobile communications*. 2014, p. 289, ISBN: 1608076997. [Online]. Available: https://books.google.pt/books/about/Introduction_to_4G_Mobile_Communications.html?id=lutPAwAAQBAJ&redir_esc=y.

[46] R. Kottath, P. Narkhede, V. Kumar, V. Karar, and S. Poddar, "Multiple model adaptive complementary filter for attitude estimation", *Aerospace Science and Technology*, vol. 69, pp. 574–581, 2017, ISSN: 1270-9638. DOI: 10.1016/J.AST.2017.07.011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963816303418.

[47] E. Kristiansen, Claude Loisy, and W. V. D. Bosch, "Road traffic monitoring by satellite", *Technical & Operational Support*, 2003. [Online]. Available: http://www.esa.int/esapub/bulletin/bullet115/chapter7_bul115.pdf.

[48] V. Lampkin and International Business Machines Corporation. International Technical Support Organization., *Building Smarter Planet solutions with MQTT and IBM WebSphere MQ Telemetry*. IBM, 2012, p. 249, ISBN: 0738437085.

[49] I. Leontiadis, G. Marfia, D. Mack, G. Pau, C. Mascolo, and M. Gerla, "On the effectiveness of an opportunistic traffic management system for vehicular networks", *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, vol. 12, no. 4, 2011. DOI: 10.1109/TITS.2011.2161469. [Online]. Available: http://ieeexplore.ieee.org/ielx5/6979/6082048/05970119.pdf?tp=&arnumber=5970119&isnumber=6082048.

[50] S. Li, L. D. Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and internet of things", *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2177–2186, Nov. 2013, ISSN: 1551-3203. DOI: 10.1109/TII.2012.2189222. [Online]. Available: http://ieeexplore.ieee.org/document/6159081/.

[51] Libelium, *Lorawan technology for arduino, waspmote and raspberry pi*. [Online]. Available: https://www.cooking-hacks.com/documentation/tutorials/lorawan-for-arduino-raspberry-pi-waspmote-868-900-915-433-mhz/.

[52] ——, *Vehicle traffic monitoring platform with bluetooth sensors over zigbee*, 2011. [Online]. Available: http://www.libelium.com/vehicle_traffic_monitoring_bluetooth_sensors_over_zigbee/.

[53] LoRa Alliance, *Lora technology*. [Online]. Available: https://www.lora-alliance.org/What-Is-LoRa/Technology.

[54] J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni, "A disruption tolerant architecture based on mqtt for iot applications", in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, Jan. 2017, pp. 71–76, ISBN: 978-1-5090-6196-9. DOI: 10.1109/CCNC.2017.7983084. [Online]. Available: http://ieeexplore.ieee.org/document/7983084/.

[55] Mark Williams, *Create a digital compass with the raspberry pi*, 2015. [Online]. Available: http://ozzmaker.com/compass2/.

[56] N. Naik and P. Jenkins, "Web protocols and challenges of web latency in the web of things", in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, IEEE, Jul. 2016, pp. 845–850, ISBN: 978-1-4673-9991-3. DOI: 10.1109/ICUFN.2016.7537156. [Online]. Available: http://ieeexplore.ieee.org/document/7537156/.

[57] K. Nellore and G. Hancke, "A survey on urban traffic management system using wireless sensor networks", *Sensors*, vol. 16, no. 2, p. 157, Jan. 2016, ISSN: 1424-8220. DOI: 10.3390/s16020157. [Online]. Available: http://www.mdpi.com/1424-8220/16/2/157.

[58] S. Oh, S. Ritchie, and C. Oh, "Real-time traffic measurement from single loop inductive signatures", *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1804, pp. 98–106, Jan. 2002, ISSN: 0361-1981. DOI: 10.3141/1804-14. [Online]. Available: http://trrjournalonline.trb.org/doi/10.3141/1804-14.

[59] J. Olsson, "6lowpan demystified", 2014. [Online]. Available: http://www.ti.com/lit/wp/swry013/swry013.pdf.

[60] J. Paefgen, F. Kehr, Y. Zhai, and F. Michahelles, "Driving behavior analysis with smartphones: insights from a controlled field study", *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, p. 36, 2012. DOI: 10.1145/2406367.2406412. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2406367.2406412.

[61] H. Park, H. H. Lee, and S.-H. Lee, "Ieee 802 standardization on heterogeneous network interworking", *International Conference on Advanced Communication Technology*, 2014. [Online]. Available: http://ieeexplore.ieee.org/ielx7/6767438/6778899/06779137.pdf?tp=&arnumber=6779137&isnumber=6778899.

[62] A. J. Paul, N. Chilamkurti, A. Daniel, and S. Rho, *Intelligent vehicular networks and communications : fundamentals, architectures and solutions*. 2016, p. 227, ISBN: 9780128095461. [Online]. Available: https://books.google.pt/books?hl=pt-PT&lr=&id=RybjCwAAQBAJ&oi=fnd&pg=PP1&dq=vehicular+networks+solutions&ots=ClJl2nBGTl&sig=Byp3Hk7uTz9bxnL5ooQJ8M_j6-k&redir_esc=y#v=onepage&q=vehicular%20networks%20solutions&f=false.

[63] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: a survey", *IEEE Communications Surveys & Tutorials*, vol. 16, no.

1, pp. 414–454, 2014, ISSN: 1553-877X. DOI: 10.1109/SURV.2013.042313.00197. [Online]. Available: http://ieeexplore.ieee.org/document/6512846/.

[64] ——, "Sensing as a service model for smart cities supported by internet of things", *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, Jan. 2014, ISSN: 21613915. DOI: 10.1002/ett.2704. [Online]. Available: http://doi.wiley.com/10.1002/ett.2704.

[65] Pieter-Jan, *Reading a imu without kalman: the complementary filter | pieter-jan.com*, 2013. [Online]. Available: http://www.pieter-jan.com/node/11.

[66] I. Poole, *Lora wireless | lora long range m2m iot*, 2015. [Online]. Available: http://www.radio-electronics.com/info/wireless/lora/basics-tutorial.php.

[67] S. P. Raut and A. Pachghare, "Advanced traffic management system (atms) for reducing congestion and collisions on road", vol. 2, no. 2, pp. 360–362, 2016. [Online]. Available: https://pdfs.semanticscholar.org/e8fa/2956d7e659608840af23942f5ab23da6cc7a.pdf.

[68] P. Reinartz, M. Lachaise, E. Schmeer, T. Krauss, and H. Runge, "Traffic monitoring with serial images from airborne cameras", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 61, no. 3-4, pp. 149–158, Dec. 2006, ISSN: 0924-2716. DOI: 10.1016/J.ISPRSJPRS.2006.09.009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271606001146.

[69] C. C. Rolim, P. C. Baptista, G. O. Duarte, and T. L. Farias, "Impacts of on-board devices and training on light duty vehicle driving behavior", *Procedia - Social and Behavioral Sciences*, vol. 111, pp. 711–720, Feb. 2014, ISSN: 1877-0428. DOI: 10.1016/J.SBSPRO.2014.01.105. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877042814001062.

[70] F. Seraj, K. Zhang, O. Turkes, N. Meratnia, and P. J. M. Havinga, "A smartphone based method to enhance road pavement anomaly detection by analyzing the driver behavior", in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers - UbiComp '15*, New York, New York, USA: ACM Press, 2015, pp. 1169–1177, ISBN: 9781450335751. DOI: 10.1145/2800835.2800981. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2800835.2800981.

[71] Spectrummonitoring, *List of mobile frequencies by country (gsm, cdma, umts, lte)*, 2017. [Online]. Available: http://www.spectrummonitoring.com/frequencies/#Portugal.

[72] R. Tadayoni and A. Henten, "From ipv4 to ipv6: lost in translation?", *Telematics and Informatics*, vol. 33, no. 2, pp. 650–659, 2016. DOI: 10.1016/j.tele.2015.10.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736585315001240.

[73] S. Taneja and P. R. Gupta, "Python as a tool for web server application development", *International Journal of Information, Communication and Computing Technology*, pp. 2347–7202, 2014. [Online]. Available: https://www.jimsindia.org/8i_Journal/VolumeII/Python-as-a-tool-for-web-server-application-development.pdf.

[74] N. Tantitharanukul, K. Osathanunkul, K. Hantrakul, P. Pramokchon, and P. Khoenkaw, "Mqtt-topic naming criteria of open data for smart cities", in *2016 International Computer Science and Engineering Conference (ICSEC)*, IEEE, Dec. 2016, pp. 1–6, ISBN: 978-1-5090-4420-7. DOI: 10.1109/ICSEC.2016.7859892. [Online]. Available: http://ieeexplore.ieee.org/document/7859892/.

[75] R. Toomey, *Zelt vehicle monitoring*, 1999. [Online]. Available: https://www.traffictechnology.co.uk/vehicle-monitoring/zelt-loop-detection?showall=1.

[76] T. Tsubota, A. Bhaskar, E. Chung, and R. Billot, "Arterial traffic congestion analysis using bluetooth duration data", *Faculty of Built Environment and Engineering; School of Urban*

Development; Smart Transport Research Centre, 2011. [Online]. Available: http://eprints.qut.edu.au/46312/.

[77] R. Vaiana, T. Iuele, V. Astarita, M. V. Caruso, A. Tassitani, C. Zaffino, and V. P. Giofrè, "Driving behavior and traffic safety: an acceleration-based safety evaluation procedure for smartphones", *Modern Applied Science*, vol. 8, no. 1, p. 88, 2014. DOI: 10.5539/mas.v8n1p88. [Online]. Available: http://www.ccsenet.org/journal/index.php/mas/article/view/29037.

[78] M. Vuagnoux and S. Pasini, "An improved technique to discover compromising electromagnetic emanations", in *2010 IEEE International Symposium on Electromagnetic Compatibility*, IEEE, Jul. 2010, pp. 121–126, ISBN: 978-1-4244-6305-3. DOI: 10.1109/ISEMC.2010.5711257. [Online]. Available: http://ieeexplore.ieee.org/document/5711257/.

[79] T. J. Wallington, J. L. Sullivan, and M. D. Hurley, "Emissions of co2, co, nox, hc, pm, hfc-134a, n2o and ch4 from the global light duty vehicle fleet", *Meteorologische Zeitschrift*, vol. 17, no. 2, pp. 109–116, Apr. 2008, ISSN: 0941-2948. DOI: 10.1127/0941-2948/2008/0275. [Online]. Available: http://www.schweizerbart.de/papers/metz/detail/17/56618/Emissions_of_CO2_CO_NOx_HC_PM_HFC_134a_N2O_and_CH4?af=crossref.

[80] YuFang Dan and Zhongshi He, "A dynamic model for urban population density estimation using mobile phone location data", in *2010 5th IEEE Conference on Industrial Electronics and Applications*, IEEE, Jun. 2010, pp. 1429–1433, ISBN: 978-1-4244-5045-9. DOI: 10.1109/ICIEA.2010.5514844. [Online]. Available: http://ieeexplore.ieee.org/document/5514844/.

[81] S. Zamfir, T. Balan, I. Iliescu, and F. Sandu, "A security analysis on standard iot protocols", 2016. [Online]. Available: http://ieeexplore.ieee.org/ielx7/7738604/7754593/07754665.pdf?tp=&arnumber=7754665&isnumber=7754593.

[82] I. P. Zarko, K. Pripuzic, M. Serrano, and M. Hauswirth, "Iot data management methods and optimisation algorithms for mobile publish/subscribe services in cloud environments", in *2014 European Conference on Networks and Communications (EuCNC)*, IEEE, Jun. 2014, pp. 1–5, ISBN: 978-1-4799-5280-9. DOI: 10.1109/EuCNC.2014.6882657. [Online]. Available: http://ieeexplore.ieee.org/document/6882657/.

[83] J. Zhang, Y. Lu, Z. Lu, C. Liu, G. Sun, and Z. Li, "A new smart traffic monitoring method using embedded cement-based piezoelectric sensors", *Smart Materials and Structures*, vol. 24, no. 2, p. 025 023, Feb. 2015, ISSN: 0964-1726. DOI: 10.1088/0964-1726/24/2/025023. [Online]. Available: http://stacks.iop.org/0964-1726/24/i=2/a=025023?key=crossref.6aef7f6cd4ae22511ff01e9dd588b0f5.

[84] S. Zhang, Y. Wu, H. Liu, R. Huang, P. Un, Y. Zhou, and L. Fu, "Real-world fuel consumption and co2 (carbon dioxide) emissions by driving conditions for light-duty passenger vehicles in china", *Energy*, vol. 69, pp. 247–257, May 2014, ISSN: 0360-5442. DOI: 10.1016/J.ENERGY.2014.02.103. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544214002503.

[85] Q. Zhao, Y. Nakamoto, and Z. Hussin, "Energy-efficient protocol for extending battery life in wireless sensor networks", in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, IEEE, Jul. 2013, pp. 268–273, ISBN: 978-1-4799-3248-1. DOI: 10.1109/ICDCSW.2013.8. [Online]. Available: http://ieeexplore.ieee.org/document/6679899/.